2016-03-25

Sparse Model Learning for Identifying Nucleotide Motifs and Inferring Genotype and Phenotype Associations

Indika Priyantha Kuruppu Appuhamilage
University of Miami, indika@miami.edu

UNIVERSITY OF MIAMI

SPARSE MODEL LEARNING FOR IDENTIFYING NUCLEOTIDE MOTIFS AND INFERRING GENOTYPE AND PHENOTYPE ASSOCIATIONS

By

Indika P. Kuruppu Appuhamilage

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

May 2016

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

SPARSE MODEL LEARNING FOR IDENTIFYING NUCLEOTIDE MOTIFS AND INFERRING GENOTYPE AND PHENOTYPE ASSOCIATIONS

Indika P. Kuruppu Appuhamilage

Approved:

_____ Xiaodong Cai, Ph.D. Professor of Electrical and Computer Engineering	_____ Miroslav Kubat, Ph.D. Associate Professor of Electrical and Computer Engineering
_____ Jie Xu, Ph.D. Assistant Professor of Electrical and Computer Engineering	_____ Sawsan Khuri, Ph.D. Assistant Professor of Computer Science
_____ Stefan Wuchty, Ph.D. Associate Professor of Computer Science	_____ Guillermo Prado, Ph.D. Dean of the Graduate School

KURUPPU APPUHAMILAGE, INDIKA P. (Ph.D., Electrical and Computer Engineering)

Sparse Model Learning for Identifying Nucleotide Motifs and Inferring Genotype and Phenotype Associations (May 2016)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Xiaodong Cai.
No. of pages in text. (191)

Variation in gene expression is an important mechanism underlying phenotypic variation in morphological, physiological and behavioral traits as well as disease susceptibility. A connection between DNA variants and gene expression levels not only provides more understanding of the biological network, but also enhances the mapping of these quantitative traits. Thus, an understanding of the mechanism of gene expression and the genotype/phenotype relationship is of paramount importance to both scientific research and social economics.

The primary functionality of the gene expression process is to convert information stored in genes into gene products such as RNAs or proteins. The fundamental of this complex process is controlled by a class of proteins known as transcription factors (TFs) that bind to special locations of the DNA double helix. These special binding sites, known as transcription factor binding sites (TFBSs), are generally short motifs of 6-20 base pairs. Furthermore, the discovery of new TFBSs will contribute to the establishment of gene regulation networks, diagnosis of genetic diseases and new drug design.

On the other hand, the genotype/phenotype relationship is mainly explained by multiple quantitative trait loci (QTLs), epistatic effects and environmental factors. A QTL is a section of DNA that correlates with variation in a phenotype. The QTL typically is linked

to, or contains, the genes that control that phenotype interactions among QTLs or between genes, and environmental factors contribute substantially to variation in complex traits. During the last two decades the use of QTLs has proven to be effective for increasing food production, resistance to diseases and pests, tolerance to heat, cold and draught, and to improve nutrient content in animal and plant breeding.

Therefore, the objective of this dissertation is to develop sparse models for such high dimensional data, develop accurate sparse variable selection and estimation algorithms for the models and design statistical methods for robust hypothesis tests for the TFBSs identification and QTL mapping problems. Although the sparse model learning works presented in this thesis are used in the context of TFBSs identification or QTL mapping problems, the algorithms are equally applicable to a broad range of problems, such as whole-genome QTL mapping and pathway-based genome-wide association study (GWAS), etc.

The widely used computational methods for identifying TFBSs based on the position weight matrix (PWM) assume that the nucleotides at different positions of the TFBSs are independent. However, several experimental results demonstrate the dependencies among different positions. Recently, Bayesian networks (BN) and variable order Bayesian networks (VOBN) were proposed to model such dependencies and thereby improve the accuracy of predicting TFBSs. However, BN and VOBN model the dependencies in a directional manner, which may hinder their capability of completely capturing complex dependencies. To this end, we develop a Markov random field (MRF) based model for TFBSs capable of capturing complex unidirectional relationships among motifs. To capture the large extent of dependencies in a sparse model without causing overfitting, we

develop a feature selection method that carefully chooses only the most relevant features of the model.

An exhaustive simulation study affirmed that our MRF-based method outperforms other state-of-the-art methods based on VOBN. To further reduce the computational complexity of our algorithm, we introduce a novel pairwise MRF model to the TFBSs, and develop a fast algorithm to learn the model parameters. Specifically, we adopt an optimization method that employs the log determinant relaxation approach to evaluate the partition function in the MRF, which dramatically reduces the computational complexity of the algorithm.

For the genotype/phenotype association problem, we develop a novel empirical Bayesian least absolute shrinkage and selection operator (EBlasso) algorithm with normal and exponential (NE) and normal, exponential and gamma (NEG) hierarchical prior distributions. Both of these algorithms employ a novel proximal gradient approach to simultaneously estimate model parameters that leads to extremely fast convergence. Furthermore, we develop a novel proximal gradient hybrid model capable of detecting more QTLs than its vanilla flavor, but still maintaining a lower false positive rate.

Having both covariance and posterior modes estimated, they also provide a statistical testing method that considers as much information as possible without increasing the degrees of freedom (DF). Extensive simulation studies are carried out to evaluate the performance of the proposed methods, and real datasets are analyzed for validation. Both simulation and real data analyses suggest that the new methods are fast and accurate genotype-phenotype association methods that can easily handle high dimensional data, including possible main and interaction effects with orders of magnitude faster than

existing state-of-the-art methods. Specifically, with the EBlasso-NEG, our new algorithm could easily handle more than 10^5 possible effects within few seconds running on an average personal computer.

Given the fundamental importance of gene expression and genotype/phenotype associations in understanding the genetic basis of complex biological system, the MRF, pairwise-MRF, EBlasso-NE, EBlasso-NEG and EBlasso-NEG hybrid algorithms and software packages developed in this dissertation achieve the effectiveness, robustness and efficiency needed for successful application to biology. With the advancement of high-throughput molecular technologies in generating information at genetic, epigenetic, transcriptional and posttranscriptional levels, the methods developed here have broad applications to infer TFBSs and different types of genotype and phenotypes associations.

.

To my family

Acknowledgments

This dissertation would not have been possible without the help of so many people in so many ways. I would like to express my sincere gratitude first to my advisor and chairman of the committee, Professor Xiaodong Cai, for his continuous suggestions, guidance, and support throughout my Ph.D. studies. I am also thankful to my committee members, Professor Miroslav Kubat, Professor Jie Xu of the Electrical and Computer Engineering Department, and Professor Sawsan Khuri, Professor Stefan Wutchy of the Computer Science Department at University of Miami for accepting the appointment to the dissertation committee and for their helpful suggestions and support.

I wish to acknowledge Professor Reuven Lask for giving me the opportunity to work as a teaching assistant during my stay at the University of Miami. I also extend my thanks to Ms. Rosamund Coutts, Ms. Michelle Perez, Ms. Angie Del-Llano and Ms. Kendra Parks for their administrative assistance.

To all my beloved friends who have supported me throughout these years with their continuous encouragement, considerations and assistance, I offer my sincerest thanks. Finally, I would like to extend my utmost gratitude to my parents, Somalatha

Grace and Wijayasinghe; my brother, Chaminda Saman Kumara; my grandmother Seelawathi Perera; my wife Disna Ranasinghe; my daughter Sayuri Kuruppu and my son Sanuj Kuruppu, for their caring, encouragement and love, which made this work possible.

INDIKA P. KURUPPU APPUHAMILAGE

University of Miami

May 2016

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

A gene is the molecular unit of heredity that is a region (locus) of deoxyribonucleic acid (DNA) that encodes a functional ribonucleic acid (RNA) or protein product [Sla14, AJL$^+$14]. The individual organism exhibits observable characteristics known as traits or phenotypes that may be quantitative, such as color of the skin, height, or qualitative, such as blood type, risk for specific diseases. The transmission of genes to an organism's offspring is the basis of the inheritance of phenotypic traits. Most biological traits are under the influence of polygenes (many different genes) as well as gene-gene and gene-environment interactions.

Genes can acquire mutations in their sequence, leading to different variants, known as alleles, in the population. An allele is one of the possible forms of a gene at a given position. These are distinguished by four types of bases in the double DNA strands: adenine (A), cytosine (C), guanine (G) and thymine (T). These alleles encode slightly different versions of a protein, which cause different phenotype traits. Genes evolve due to natural selection or survival of the fittest of the alleles.

The concept of a gene continues to be refined as new phenomena are discovered [GH06]. For example, regulatory regions of a gene can be far removed from

its coding regions, and coding regions can be split into several exons. Some viruses store their genome in RNA instead of DNA, and some gene products are functional non-coding RNAs. Therefore, a broad, modern working definition of a gene is any discrete locus of heritable genomic sequence that affects an organism's traits by being expressed as a functional product or by regulation of gene expression [Pea06, Pen07].

A great number of genes are differentially expressed among individuals. The variation in gene expression is a more important mechanism underlying phenotypic variation in morphological, physiological and behavioral traits as well as disease susceptibility. A connection between DNA variants and gene expression levels not only provides more understanding of the biological network but also enhances the mapping of the above-mentioned quantitative traits. In fact, gene expression levels themselves can be treated as quantitative traits.

Thus, an understanding of the mechanism of gene expression and the relationship of genotype to phenotype is of paramount importance to both scientific research and social economics. The primary functionality of the gene expression process is to convert information stored in genes into gene products such as RNAs or proteins. Since all cells contain the same information (with some exceptions), it is essential for an organism to increase versatility and adaptability by allowing cells to express different proteins when needed. As an example, in the course of embryonic development, a fertilized egg cell gives rise to many cell types that differ dramatically in both structure and function [ABH$^+$03]. The fundamental of this complex process is controlled by a class of proteins known as transcription factors (TFs) [Wat04] that bind to special locations of DNA double helix. These special binding sites are known as transcription

factor binding sites (TFBSs), which are generally short motifs of length in the range of 6-20 base pairs [BN03, JLZL04, PMP04]. Other than an understanding of gene regulation mechanism, the discovery of new TFBSs will contribute to the establishment of gene regulatory networks, diagnosis of genetic diseases, new drug design, etc.

On the other hand, the genotype/phenotype relationship is mainly explained by multiple quantitative trait loci (QTLs), epistatic and environmental factors. QTL is a section of DNA that correlates with variation in a phenotype (the quantitative trait). QTL typically is linked to, or contains, the genes that control that phenotype Interactions among QTLs or between genes, and environmental factors make a substantial contribution to variation in complex traits [CH04]. Use of quantitative trait loci (QTLs) has proven to be an effective tool to increase food production, resistance to diseases and pests, tolerance to heat, cold and draught, and to improve nutrient contents in animal and plant breeding during the last two decades [BBC08]. Thus, the identification of transcription factor binding sites and QTL mapping studies opens up two major avenues to the research community and still remains as active research topic. First, we briefly sketch the biological background.

1.1 Biological Background

Gregor Johann Mendel, considered as the father of genetics, was the first to discover the existence of biological elements called genes in 1866 [Men65] by conducting experiments with pea plants. The significance of his work was not recognized until it was rediscovered by Hugo de Vries and Carl Correns in 1900 [Bow03]. Swiss physiological

chemist Friedrich Miescher is considered as the discoverer of deoxy-ribonucleic acid (DNA), which he called nuclein (now nucleic acids) inside the nuclei of human white blood cells in 1869 [Pra08]. Later, in 1953, James Watson and Francis Crick discovered the double-helical, three-dimensional structure of the DNA molecule [FG53, Cri74].

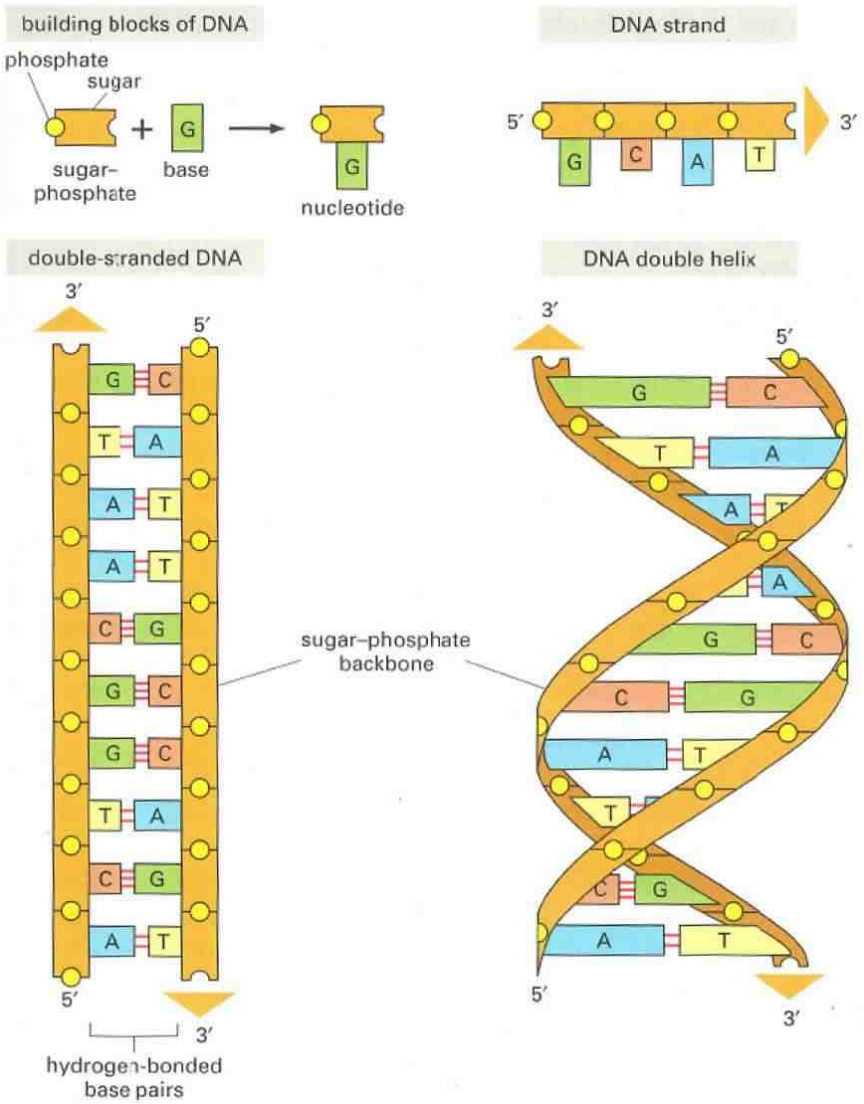

Figure 1.1: DNA is made of four nucleotide building blocks [ABH+03]

As described in Figure 1.1, DNA is composed of four basic components, namely, nucleotides, which are identical except that each contains a different nitrogen base.

Each nucleotide contains a phosphate, a sugar and one of the four bases: Adenine, Guanine, Cytosine, and Thymine usually denoted as A, G, C, and T, respectively. The double-helical structure is made by combining these bases as pairs with hydrogen bonds. Each base pairs consists of A and T or C and G. Due to the asymmetrical structure of sugars, the DNA molecule is directional. Each sugar is connected to the strand upstream in its fifth carbon and to the strand downstream in its third carbon. Therefore, in biological jargon, the DNA strand goes from 5' to 3' as shown in Figure 1.1.

The complete set of information contained within an organism is called its genome. For simplicity, we can think of this information as being written as a long chain of four-letter alphabet: A, T, G and C. This extremely long double-stranded DNA is compacted into chromosomes (see Figure 1.2) in eukaryotes (a higher level organism whose cells have nuclei). Although a similar name appears as bacterial chromosome in prokaryotic (single cell organisms whose cell does not have nuclei) cells, the structure is different. In eukaryotic organisms, every cell, with few exceptions, has a complete genome set.

Information coded in the genome leads to the synthesis of proteins. A protein is an amino acid sequence (a molecule containing both amine and carboxyl functional groups) and considered to be the principal constituent of cells that determines their structure and functionality. Each protein has its own unique amino acid sequence, usually called polypeptide chain, that dictates the protein to have a distinctive shape and chemical properties. This complex process of transforming information in genome to produce proteins is known as gene expression.

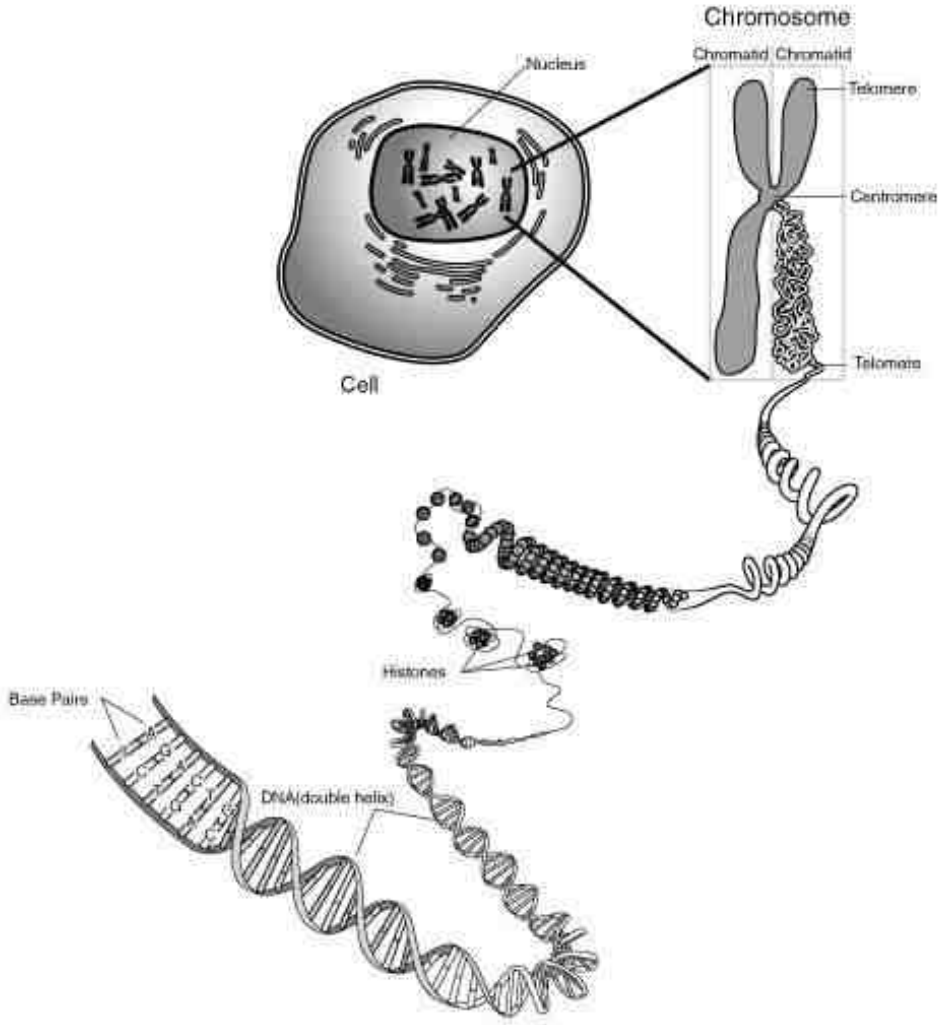

Figure 1.2: DNA is compacted to form chromosome

This process mainly consists of two steps: transcription and translation. Transcription refers to the process of copying information in DNA to form ribonucleic acid (RNA). RNA is a single stranded linear chain of polymer made of four different types of nucleotide subunits linked together by phosphodiester bonds. There are two main differences between DNA and RNA: (1) the nucleotides in RNA are ribonucleotides; (2) RNA has uracil (U) in one of the bases, instead of the thymine(T) found in DNA. Therefore, we can assume that the RNA has the same information written with alphabet A, G, C and U. The process of using information contained in RNA to form protein is known as translation.

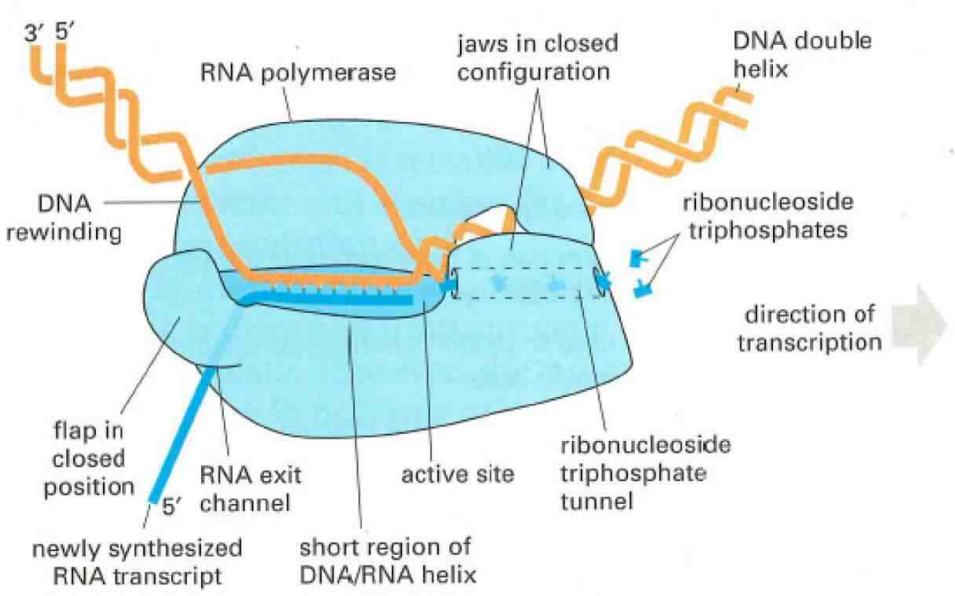

Figure 1.3: Transcription process [ABH+03]

The enzymes that carry out the transcription process are known as RNA polymerases. Even though bacteria contain only a single type of RNA polymerase, eukaryotic cells have three types: RNA polymerase I, RNA polymerase II, and RNA polymerase III. Out of these three, the vast majority of genes, including all those

that encode proteins, are transcribed by RNA polymerase II. The RNA polymerase moves stepwise along the DNA, unwinding the DNA helix just ahead to expose a new region as the template strand for complementary base-pairing as shown in Figure 1.3. Therefore, the growth of the RNA chain is extended from 5' to 3' direction.

Even though all cells in multicellular organisms have the same genetic information (with some exceptions), different cell types produce different sets of proteins at different times. This is due to the controlling mechanism regulating gene expression. Controlling the gene expression is a very complex process and eukaryotic cells have more complex controlling mechanisms which are not available to bacterial cells. As discussed earlier, the DNA-to-protein conversion process begins with the transcription process. Therefore, the factors that initiate the transcription process have larger impact over the controlling mechanism of gene expression.

Bacterial RNA polymerase is able to initiate transcription without the help of additional proteins. However, eukaryotic RNA polymerases need the help of a large set of proteins called general transcription factors (TFs). These must assemble at each promoter (promoter is a region of DNA that facilitates the transcription of a particular gene that typically located near the genes they regulate) along with the polymerase before the polymerase can begin transcription. These special binding sites in the DNA are called transcription factor binding sites (TFBSs). Together with TFs, these binding sites position the RNA polymerase correctly at the promoter, to aid in pulling apart the two strands of DNA to allow transcription to begin. This also allows RNA polymerase to leave the promoter as transcription begins. Usually, the length of these short TFBSs ranges from 6 to 20 base pairs [BN03, JLZL04, PMP04].

Moving back to the genotype/phenotype associations, Mendel's principles of heredity explain qualitative traits that are controlled by single gene locus, while quantitative inheritance theory assumes that phenotypes result from multiple gene factors, gene-gene interactions as well as environmental effects [FM96]. Each of the main and interaction effects exhibits only a modest effect on the phenotype and it is difficult to dissect their individual effect. The genotype/phenotype association in animal/plant breeding can be modeled by QTL mapping in inbred lines. In human populations, this relationship is studied by examining the association of phenotypes with the natural occurring genetic variations such as SNPs.

With the advent of new DNA sequencing technologies, high density markers can be easily generated along the genome. However, it is still very likely that true causal markers are not captured due to the large amount of genomic variants in living organisms [AJL$^+$14, Con07]. On the other hand, with the large number of available genetic markers, researchers usually have a hard time in finding the number, location and effect of the individual genetic marker involved in the inheritance of target phenotypes. Therefore, the correlation among genetic markers and oversaturated models are two common properties in QTL mapping.

Now that we have provided a basic biological background, the rest of the discussion is devoted to further exploring the two avenues we described earlier: TFBSs identification and the QTL mapping.

1.2 TFBSs Identification

A large number of methods have been proposed toward identification of TFBSs due to the increased interest in this area. In general, these methods can be categorized as experimental and computational.

1.2.1 Experimental Methods to Identify TFBSs

Experimental methods are the traditional way of identifying TFBSs through laboratory experiments. These methods are usually known as wet-lab experiments, and there are a number of popular methods such as DNAse I protection or footprinting assays [ZG03] and electrophoretic mobility shift assay (EMSA) [GR81]. Due to scale problems, though, these methods are not best suited for whole genome-wide analysis. However, some high-throughput methods also have been developed, such as systematic evolution of ligands by exponential enrichment (SELEX) [TG90] and protein-binding DNA microarray. The most common high-throughput technique is chromatin immunoprecipitation of bound DNA followed by either microarray hybridization (ChIP-chip) or sequencing (ChIP-seq) [Bul03, RRW$^+$00]. These methods are generally time consuming and expensive. Therefore, various computational methods have emerged due to the advancement of computer technologies with much lower cost.

1.2.2 Computational Methods for Identification of TFBSs

There are two basic approaches to model and identify TFBSs [JLZL04] using computational methods. One is to represent a specific TFBS as a consensus sequence and use a search algorithm to find the sequences that match the most positions of the consensus sequence, allowing one or several mismatches. The other approach is to model a TFBS using a statistical model and employ a statistical method to search for sequences that are statistically consistent with the model. With both these categories, the algorithms that do not use any training data sets to identify putative binding sites are called *de novo* methods. These methods usually search over promoter regions of a number of genes that have common TFBSs that would appear more than expected from chance alone. Let us first briefly describe the nature of each type of algorithms before we go into more detail.

1.2.3 Statistical Methods over Consensus Methods

Consensus-based methods represent a motif as a sequence of characters defined by the International Union of Pure and Applied Chemistry (IUPAC) [IUP86] as shown in Table 1.1. For example, $GAL4$ binding sites in yeast can be represented as CGGNNNNNNNNNNCCG [ZZ99] where N stands for any nucleotide. Thus, any oligos (short term for oligonucleotide: a short sequence of nucleotide) starting with CGG and ending with CCG, with any 11 nucleotides in the middle, are considered as GAL4 binding sites. These methods usually belong to the class of string-matching problems

where some distance measure such as *Levenshtein* distance is used to calculate the distance between pattern and the sequence being considered.

Table 1.1: IUPAC codes for Nucleic acids

code	description
A	Adenine
C	Cytosine
G	Guanine
T	Thymine
U	Uracil
R	Purine (A or G)
Y	Pyrimidine (C, T, or U)
M	C or A
K	T, U, or G
W	T, U, or A
S	C or G
B	C, T, U, or G (not A)
D	A, T, U, or G (not C)
H	A, T, U, or C (not G)
V	A, C, or G (not T, not U)
N	Any base (A, C, G, T, or U)

The first consensus-based method to identify TATA-box motif was developed by Galas et al. [GES85]. Since then, many consensus based methods have emerged [EP02, ST02, PMMP04]. However, the main problem with these methods is the omission of important variable regions of binding sites [RFJ$^+$98] and inability of providing information of the relative frequency of alternative nucleotides. Therefore, statistical-based methods have become popular over the consensus based methods because the statistical approach is flexible in handling missing data and incorporating information

from various sources. A few of the most popular TFBS-finding algorithms are all based on explicit statistical modeling [JLZL04].

A widely used statistical model is based on the position frequency matrix, which contains the frequency of each nucleotide at each position of TFBS, or equivalently the position weight matrix (PWM) [KGR+03] that contains the probability of each nucleotide at each position of the TFBS. PWM is also known as position-specific scoring matrix (PSSM). In fact, TF databases such as TRANSFAC [MFG+03] and the SCPD [ZZ99] use this representation to describe binding site consensuses. The PWM model has been successfully applied not only to the problem of TFBS identification, but also to other diverse problems in DNA and protein sequences analysis [Sal97].

Although some other scoring models have been used to improve the performance over the PWM, they are not as prevalent as the simple PWM model [FH97] in identification of TFBSs. Some other statistical methods have been proposed to overcome certain drawbacks in the PWM based methods. We will discuss these in more detail in the next section.

1.2.4 Motivation for the TFBSs Identification Problem

PWM-based methods [SSGE82, HHG83] assume statistical independence among base pairs at different positions. Therefore, the joint probability of a given sequence segment factorizes into the product of single position probabilities [DSS03, EG01]. However, there has been increasing evidence suggesting the existence of correlations between nucleotide positions among TFBSs [BJC02, BLFS01, WGRP99, MS01,

BPQ+06, UMFK02, OPLS05]. Man and Stormo [MS01] showed that the interaction of salmonella bacteriophage repressor Mnt with its operator DNA at position 16 and 17 was not independent. Furthermore, dependencies within binding sites of Arabidopsis ABA responsive element were described by Barash et al. [BEFK03] and Udalova et al. [UMFK02] demonstrated the existence of such interdependent effects in binding sites of the NF-k B protein. Therefore, PWM based methods often result in a high rate of false-positive predictions [TLB+05, JCIL05].

Although dependencies between a base pair and several immediate upstream or downstream base pairs are captured to a certain extent by fixed order Markov models and hidden Markov models (HMMs) [TLM+01, LBL01, OH99, ON01], they fail to represent distant relationships among base pairs. In fact, these long distance dependencies can still be modeled with higher order Markov models, but the large number of variables that increases exponentially with the order of the model leads to over-fitted models as well as prohibitive amount of computation. To alleviate such problems, Bayesian networks (BN) were proposed to identify TFBSs. The BN model is a directed graphical representation of probabilistic dependence knowledge [Pea88], where the edges are directed from an influencing position (parent) into an influenced position (child). BN model has been applied to a vast number of applications such as analysis of gene expression data [FLNP00], genetic linkage analysis [HGC95] and the identification of TFBSs and other functional DNA regions [BEFK03, CDKK00, CG04]. Unlike Markov models, BN models do not assume that the dependencies are necessarily between adjacent positions.

To further reduce the number of parameters and the likelihood of over-fitting, variable order Bayesian networks (VOBN) were proposed to model TFBSs [BGSG+05]. The VOBN approach extends the inhomogeneous variable order Markov (VOM) model by allowing dependencies among non-adjacent positions while maintaining fewer parameters. They first construct a BN model, where a node corresponds to a position, and a directed edge (an arrow) from node i to node j meaning that the nucleotide at position i is considered as part of the context of the nucleotide at position j. After that, they created a VOM tree for each position of the sequence and employed a pruning method by using *Kullback-Leibler* (KL) divergence [Kul59] of the conditional probabilities of symbols between each leaf and its parent node. However, both BN and VOBN are based on directed graphs. This imposes some restrictions on modeling dependencies and may not completely capture the dependencies among base pairs. There is thus a strong motivation to develop a framework that could handle these undirected relations among base pairs still keeping the computational tractability.

1.2.5 Markov Random Field Framework for Identifying TFBSs

To solve the TFBSs identification problem, we propose using *Markov random field* (MRF) or *undirected graphical model* [KS80, Bis06]. MRF has a set of nodes or vertices, each corresponding to a variable or set of variables. Each node of the graph can connect with any other nodes through undirected links. These links are usually

called edges that represent relations or dependencies among variables. In general, this graph can have any complex structure, though some applications use special cases, such as chain-structured graphs and grid-structured graphs. When each node is connected to all the other nodes in the graph, it is known as a fully connected graph. It is worthwhile to understand a few terms that we will use throughout our discussion before going into more detail.

The subset of nodes where a link exists between all pairs of nodes in the subset is called a *clique*. In other words, the nodes in a clique are fully connected. Let us consider the example described in Figure 1.4. Variables that are assigned to each node are represented as x_1, x_2, x_3, x_4, x_5 and x_6. Note that we alternatively use the variable name to identify the node as well. As one can see from the graph, nodes x_4, x_5 and x_6 form a clique, because each has an edge to all the others. Furthermore, a *maximal clique* is a clique such that it is not possible to include any other nodes from the graph without ceasing to be a clique. In Figure 1.4, the set consisting of vertices $\{x_5, x_6\}$ is not a maximal clique because we can still add vertex x_4 to the set without losing the clique property. Therefore, the maximal clique contains vertices $\{x_4, x_5, x_6\}$.

With MRF, the joint distribution of variables over the nodes is represented as a product of *potential functions*. These functions are positive and real valued functions defined on maximal cliques of the graph. Unlike in directed graphical models (e.g., BN) where each factor in the joint distribution represents the conditional distribution of the corresponding variable, given the state of its parents, we do not impose any restrictions over the potential functions to have any specific probabilistic interpreta-

tions such as marginal or conditional distributions. This gives us more freedom to define any form of positive real valued potential function for the purpose.

One consequence of this fact is that we need to add a normalization factor to the joint distribution because, in general, the product of potential functions is not correctly normalized. Usually, this normalization factor is known as *partition function*. As we will see shortly, the main computational burden in training as well as in inferencing through the undirected graphical model is caused by this partition function.

Having described basic properties of MRF, and its importance over the directed models, let us present the problem of modeling and identification of TFBS with MRF. We define each position in the TFBS as a vertex or node in the MRF. To each node, we associate a random variable, which in our case takes values in a configuration space {A,T,G,C}. As we illustrate later, this enables us to easily define any complex dependencies among nucleotides with a proper set of feature functions as explained in Section 2.3. Although this representation helps to capture most of the complex dependencies among nucleotides in the TFBSs, the exponential number of model parameters makes the model unusable, due to the following two reasons:

1. Parameter estimation involves an extremely high computational complexity;

2. Limited training instances lead to an over-fitted model.

Therefore, we need to include a large number of parameters to capture dependencies, but including a large number of parameters may cause over-fitting problems. To solve this paradox, we propose to use an iterative procedure that uses the *Max-*

imum Likelihood (ML) approach together with two levels of feature reduction techniques, as we will describe more detail in Chapter 2. The first level uses a feature induction algorithm [PPL97] that carefully adds a small number of most relevant parameters to the existing model in each iteration. The second level uses a regularization mechanism with the ML approach to eliminate less relevant parameters. In particular, we propose to use two types of regularization techniques: L_1 regularization [SK03, Ng04, DPS04, LGK07] and Bayesian regularization [CT06]. As we will demonstrate in Chapter 2, combining these two feature reduction techniques helps to make the model as sparse as possible and to capture complex dependencies while keeping the computational complexity at a feasible level.

1.2.6 Pairwise MRF Model and Fast Learning Algorithm

The proposed MRF model has a general structure where feature functions are allowed to grow to higher orders in the iteration process. Thus, the maximal clique size can be any number from one to the length of the motif. This leads to a higher computational complexity in the parameter learning phase. In particular, the major computational complexity is caused by the calculation of partition function.

The computational complexity of exact evaluation of partition function is in the order of $4^N \times K$, where N is the length of the TFBS and K is the number of parameters. Previously, we proposed to use two level feature reduction techniques together with the ML approach, which effectively reduces the value of K in each iteration. Even with efficient implementation with C++, the computational complexity of the

algorithm for long motifs, e.g., 20 nucleotides, is extremely high. To tackle this problem, we propose to model TFBS using pairwise MRF where the maximal clique size is limited to two. However, we allow any two nodes to form a clique. In this way, we capture most of the dependencies among nodes while reducing the computational complexity.

Using this pairwise MRF, we develop a fast algorithm to learn the structure of the model using an approximation method to evaluate the partition function. Specifically, we adopt an optimization method that employs the log determinant relaxation approach [WJ06] to solve this problem. As we will further explain in Section 3.2, with the dual formulation of the optimization problem, the complexity of the calculation of the partition function is reduced to the order of $(4N)^3$. This will drastically reduce the overall complexity of our learning algorithm that facilitates employing our MRF model in modeling and identifying long motifs.

1.3 QTL Mapping

In previous sections, we went over the TFBSs identification problem, where we proposed a Markov random field-based model for sparse model learning. Even though the QTL mapping problem shares some similar characteristics with the TFBSs identification problem, the number of model parameters involved in QTL mapping could be extremely large. Due to the high computational complexity of Markov random field-based models for such problems, we will take a different route. Before going into more detail, let us give some insight into the QTL mapping problem.

Phenotypic traits that vary continuously are referred to as quantitative traits. Stretches of DNA containing or linked to the genes that underlie a quantitative trait are called quantitative trait loci (QTL). QTL mapping is a scientific undertaking to identify QTL using information of biological markers. Consider the case where two inbred lines with different traits of interest are chosen to cross. In this scenario, the first generation (F_1) will have identical genetic markers that show complete linkage disequilibrium (LD) (the non-random association of alleles at different loci) for genes differing between the inbred lines. Starting from the F1, we can use several designs to study QTL mapping.

For example, intercross design (IC) is to cross between siblings among F_1 individuals. Crossing of F_1 individuals to one of the two parental lines leads to backcross design (BC). The doubled haploid (DH) design is to develop individuals from pollens of an F_1 plant through another culture and chromosome doubling; and recombinant inbred lines design (RIL) is to cross between sibling individuals for many generations starting from F_2 till almost all of the segregating loci come to be homozygous.

While the different experiment designs lead to different breeding populations for QTL mapping research, the F_2 population provides the most genetic information among different types of mapping populations [LGA$^+$09]. Even though several techniques are used in QTL mapping, such as single marker mapping, interval mapping, multiple loci mapping as well as composite interval mapping, the principles in these mapping techniques are generally the same, and methods used in one population can be extended to other experiment populations. Techniques such as simple linear regression, *t-test* and analysis of variance (ANOVA) are used by single marker test, while

multiple loci mapping considers multiple markers simultaneously that may include possible higher order marker interactions as well as environment factors.

Interval mapping [LB89] and composite interval mapping [Zen94] are extensions of single marker test and multiple loci test, respectively. In interval mapping, each locus is considered one at a time and the logarithm of the odds ratio (LOD score) is calculated for the model that the given locus is a true QTL. In composite interval mapping, one performs interval mapping using a subset of marker loci as covariates. These markers serve as proxies for other QTLs to increase the resolution of interval mapping, by accounting for linked QTLs and reducing the residual variation [LB89].

1.3.1 Statistical Methods for QTL Mapping

As biotechnology progresses, more and more biological markers become available in many organisms, and the genotyping of the markers has become less and less expensive. QTL mapping is now being used on a daily basis in genetic research as well as in medical studies. Appropriate and efficient statistical methodologies are crucial for QTL mapping to be fruitful. As the data complexity of QTL mapping evolves due to the progress of bio-technology, many advanced statistical methods have been developed in recent years.

The data points for the problem include phenotype values (quantitative such as weight and height, or qualitative including nominal values such as male/female and ordinal values such as strong/normal/weak), and genotype values (homozygote, heterozygote, or another minor homozygote in intercross and population studies). The

goal of the computational methods is to differentiate the testing markers in linkage with causal markers through genotype/phenotype association studies.

There are mainly two categories of methods: single marker analysis and interval mapping (IM). The models that take a single variant at a time are mostly used due to the less computational complexity, while the models with multiple variants are undergoing intensive research studies. We will discuss these two areas with more detail in the following sections.

1.3.2 Single Marker Analysis

Single marker analysis is simple and straightforward for most models, such as simple linear regression, analysis of variance (ANOVA), *t-test*, the Cochran-Armitage test, Pearson χ^2 test and generalized regression for qualitative traits. All these methods analyze the simplest association among genotype of markers and the phenotype quantitative trait and test for trait value differences between marker groups. The statistics tests the null hypothesis that the trait value is independent of the genotype at a particular marker, for a given trait. It rejects the hypothesis if the test statistics is larger than a threshold value. Specific to the *t-test* statistics, the hypothesis corresponds to the test of mean phenotypic values among genotypes at a marker. In regression models, the hypothesis corresponds to the test of slope.

Suppose n_1 individuals have genotype value MM at marker q in a mapping population, while n_2 have mm, and the corresponding mean phenotype values are μ_M and μ_m. The hypothesis test is to test: $H_0 : \mu_M = \mu_m$ and $H_1 : \mu_M \neq \mu_m$. Then the

t-score is calculated as $t = \frac{\mu_M - \mu_m}{\sqrt{s_p^2(1/n_1+1/n_2)}} \sim t_{n_1+n_2-2}$, where $s_p^2 = \frac{(n_1-1)s_M^2+(n_2-1)s_m^2}{n_1+n_2-2}$ with s_i^2, $i = M$ or m being the estimated phenotypic variance within the two groups, and $t_{n_1+n_2-2}$ stands for the Students t distribution with a degree of freedom (DF) equals to $n_1 + n_2 - 2$. Then the test rejects H_0 if $t > t_{\alpha/2}$, where $t_{\alpha/2}$ is a threshold defined such that $p(|t| > t_{\alpha/2}) = \alpha$, with α being a pre-specified constant typically equals to 0.05.

This can also be used to test the heterozygote genotype (a locus with different alleles) Mm against the other two genotypes to reveal dominant effects since it does not assume a specific genetic model. The ANOVA method computes a score as $F = \frac{\text{variance between M and m}}{\text{variance within M and m}} \sim F_{1,n_1+n_2-2}$, where F_{1,n_1+n_2-2} stands for the F distribution with DFs equal to 1 and $n_1 + n_2 - 2$. It rejects the null hypothesis if the score is outside of $\alpha/2 \sim (1 - \alpha/2)$ range of the F-distribution. The linear regression model assumes that the phenotype can be expressed as a linear combination of predictor plus an error term that follows a normal probabilistic distribution. The statistics tests if the regression coefficient of the linear model equals zeros, or equivalently the phenotype value equals to population mean. In the latter case, the test is equivalent to the *t-test*.

The Cochran-Armitage test evaluates the null hypothesis that the proportion of cases counts among n_1 individuals having genotype value MM, n_2 having Mm and n_3 having mm at a marker fit a straights line with zero slope [Arm55]. Pearson χ^2 test puts the genotype counts into a contingency table and computes the score as $\chi^2 = \sum_{i=1}^{p} \frac{(O_i-E_i)^2}{E_i} \sim \chi_2^2$, where p is number of cells in the table, O_i is the observed

counts in i^{th} cell, E_i is the expected counts in the given study cohort [Bal06], and χ^2_2 represents the χ^2 distribution with a DF of 2. The generalized regression model consists of three components: linear predictors, a link function and a probabilistic distribution model of the qualitative phenotype, which is typically binomial or multinomial distribution [DRW03]. Similar to linear regression, the generalized regression model for qualitative traits also tests if the regression coefficient is zero.

The methods described above use different genetic effects such as additive and dominant effect in testing genotype and phenotype associations. In regression models, they can be tested simultaneously by adding dummy variables to encode different effects. Take linear regression as an example. Suppose we observe quantitative phenotype y_i, $i = 1, \cdots, n$, of n individuals and now test marker q with possible genotype MM, Mm and mm. A widely used genetic model is the Cockerham model [Coc54] that defines the values of the additive effect as $x_i = -1$, 0 and 1 for the three genotypes and the values of the dominance effect as $z_i = -0.5$ and 0.5 for homozygotes and heterozygotes, $i = 1, 2, \cdots, n$, respectively. Then the regression model is formulized as

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 z_i + e_i,\ i = 1, 2, \cdots, n, \tag{1.1}$$

where e_i is residual error that follows a normal distribution with zero-mean and variance σ^2. The parameters are estimated by the least square estimator (LSE). For qualitative traits, the generalized linear model forms similar linear predictors:

$$\eta_i = \beta_0 + \beta_1 x_i + \beta_2 z_i,\ i = 1, 2, \cdots, n, \tag{1.2}$$

and utilizes a link function such that $E(y_i|x_i, z_i) = g^{-1}(\eta)$. The parameters are estimated by the maximum likelihood (ML) estimator.

To control the overall type I error rate, single marker methods require multiple test correction such as Bonferroni correction [Dal04] and family-wise error rate [BH95]. Given the large number of possible markers, multiple test correction leads to such a stringent criterion that most modest effects could not pass the threshold. Usually, the complex phenotype is controlled by multiple genetic factors and their interactions with environmental effects. However, each of the main and interaction effects exhibits only a modest effect on the phenotype, which makes it difficult to dissect their individual effect.

As shown in an example in prokaryotes, genes belonging to specific/non-specific membrane channels, oxidative stress response and osmotic stress response are involved in conferring bacterial resistance to high arsenic level, and looking for a single gene or single regulon may end up with no meaningful result [HTRM10]. Therefore, a marker that is identified with a single marker method as associated with the phenotype may only explain a small proportion of the phenotype variation. Before we move on to the multiple loci association methods that are capable of capturing these complex relationships, it is worthwhile understanding the interaction of genes, known as *epistatic effects*.

Epistasis refers to genetic interactions in which the mutation of one gene masks the phenotypic effects of a mutation at another locus [Cor09, Cor02, MW05]. While biological epistasis is defined as physical interaction among genes, statistical epistasis is defined mathematically as deviation from additivity in a linear model of

genotypes [MW05]. Given the nature of genes in functioning by affecting behaviors of other genetic products, epistasis is believed to play an important role in the genesis of complex disease [Ste12]. Interactions between QTLs or SNPs can be analyzed with a regression model, including two markers and their high order interaction terms [Cor02]. The two loci linear regression model is:

$$\begin{aligned} y_i = \beta_0 + \quad & \beta_{a1}x_{i1} + \beta_{d1}z_{i1} + \beta_{a2}x_{i2} + \beta_{d2}z_{i2} + \beta_{aa}x_{i1}.x_{i2} + \beta_{ad}x_{i1}.z_{i2} \\ & + \beta_{da}z_{i1}.x_{i2} + \beta_{dd}z_{i1}.z_{i2} + e_i, i = 1, 2, \cdots, n, \end{aligned} \tag{1.3}$$

where β_as are the regression coefficients for additive effects, β_ds are the regression coefficients for dominant effects, s_i and z_i are dummy variables identical to the ones in the single marker model, and the epistatic effects are modeled via element-wise product of effects in two loci. Such a two loci approach considers one epistatic effect at a time, and inherits similar multiple test problems as in the single marker approach. In fact, given the combinatorial number of possible interactions, the selection criterion is even more stringent. To mitigate the multiple test problem and to reduce the computational burden, a two-stage strategy that analyzes the subset of interactions within significant marginal effects has been proposed [ZL07, WMS+11].

1.3.3 Interval Mapping

To overcome drawbacks of single marker analysis, multiple loci association methods have been proposed to handle main and interaction effects simultaneously. The idea behind interval mapping is simple: one can gain power in testing the null hypothesis against the alternative that there is a QTL at a specified locus, by incorporating

marker information from either side of the locus. In addition to avoiding the multiple test correction problem, the multiple loci methods are also motivated by the fact that complex phenotypes are controlled by multiple genetic factors [AJX01] and that gene-gene interactions are pervasively present [Cor09].

Single variant regression can be extended to multiple-variant analysis by including multiple covariates into the regression model. It is also common to use haplotypes in LD blocks to capture the correlation structure of SNPs in regions of little recombination. A haplotype is a combination of alleles at different loci on the same chromosome [AKFST09]. The haplotype regression method leads to a model with fewer DF and captures the overall effects of tightly linked *cis*-acting causal variants [Bal06].

Despite the promise of high power and reasonable type I error rate, there are great challenges in the implementation of multiple QTLs mapping and SNPs association. First and foremost, even though high-throughput technologies are able to genotype a large number of samples efficiently, the possible number of markers p is typically far greater than the number of samples n in the study. Traditional ordinary least square regression method fails for the case $p \gg n$. When taking into account the interactions between loci, variable selection is even more demanding [WE07].

It is likely that the increased DF in a multiple QTLs model jeopardizes the power gain over single-locus models [GCS$^+$13, ZGD11]. Additionally, in binary-trait or case-control association analysis, complete or semi-complete separation can be a serious problem due to the discrete nature of both marker data and binary outcomes [Hei06, GJPS08]. Usually, the separation problem is handled by removing predictors, which, however, may result in loss of the strongest predictors [Zor05]. Therefore, accurate

variable selection methods and sparse model inference become critical for the multiple variant association approach, especially when epistasis is considered [Rit11].

1.3.4 Variable Selection in Multiple Variant Methods

It is critical to select an appropriate multi-variant model to identify correct genetic loci associated with a phenotype, particularly in the case of "large p and small n" where $p \gg n$. The objectives of designing robust variable selection techniques include accurate causal variables detection, precise estimation, as well as powerful hypothesis tests [HCMF08]. Variable selection and shrinkage are two general techniques in handling such model selection problems, and they typically produce sparse models.

Traditional variable selection methods include single variable analysis, forward selection, backward elimination and forward stagewise selection. The single variable regression test discussed in the previous section is computationally convenient for high dimensional data, but it falls short in taking interactions into consideration and requiring a stringent criterion to declare significance. Forward selection and backward elimination are regression methods that start with one or full variables and iteratively add or delete one variable to select the optimal variable subsets. The methods apply selection criteria such as the Akaike information criterion (AIC) or Bayesian information criterion (BIC) [Sch78] to penalize model complexity.

The forward stagewise method starts with an empty set and iteratively adds the variable that is most correlated with the current residual at each step, and increases the regression coefficients with a small value. These greedy selection methods may

result in a suboptimal subset, and are computationally expensive even for small number of variables [HTF09]. The variable shrinkage method includes all variables in the model and applies a penalty function or appropriate prior distributions on the variables to automatically shrink most non-effects toward zero.

Two well-known methods are the regularized regression method [Tib96, ZH05, HTF09] and the Bayesian shrinkage method [OS09]. Among them, two special cases where certain penalties such as l_1 or l_2 penalty, or hierarchical prior distributions such as Normal-Exponential-Gamma (NEG) or Normal-inverse-χ^2 distributions that favor sparseness are the most popular techniques. Consider the general multiple linear regression problem with n samples and p variables:

$$y_i = \mu + \sum_{j=1}^{p} x_{ij}\beta_j + \epsilon_i, \tag{1.4}$$

where y_i is the response variable for sample i, x_{ij} is the j^{th} predictor for the i^{th} sample, μ is the regression intercept, β_j is the regression coefficient responsible for the effect of variable X_j to y, and ϵ_i is the residual error that follows a normal distribution with zero-mean and covariance $\sigma_0^2 : \epsilon \sim N(0, \sigma_0^2)$. Defining $y = [y_1, y_2, \cdots, y_n]^T$, $\beta = [\mu, \beta_1, \beta_2, \cdots, \beta_p]^T$ and $X = [1, x_1^T, x_2^T, \cdots, x_p^T]^T$, we can write the regression model in a more compact form:

$$y = X\beta + e, \tag{1.5}$$

where e is the residual error that follows a normal distribution with zero-mean and covariance $\sigma_0^2 I : e \sim N(0, \sigma_0^2 I)$. The generic penalized regression method infers the model parameters as:

$$\hat{\beta} = \arg\min_{\beta} ||y - X\beta||_2^2 + \lambda||\eta(\beta)||_q^q, \tag{1.6}$$

where $\eta(\beta) \geq 0$, and $||\eta(\beta)||_q^q = \sum_{j-1}^{p} \eta(\beta_j)^q$ is the l_q penalty term. Here the intercept is absorbed into matrix X, and it is not penalized. Without loss of generality, variables are assumed to be rescaled to have variance 1 to obtain invariant penalties. Notice that when $\eta(\beta_j) = \begin{cases} 1, & \text{if } \beta_j \neq 0 \\ 0, & \text{otherwise} \end{cases}$, $\forall_j$ the model penalizes the number of effects.

With $q = 1$, the solution of (1.6) is equivalent to the AIC (for $\lambda = 2$) or BIC (for $\lambda = \log(n)$) in forward selection and backward elimination, and the estimates are the same as the least square solutions. Let $\eta(\beta_j) = |\beta_j|$, $j = 1, 2, \cdots, p$ to penalize the effect size, (1.6) becomes the bridge regression [FF93, Fu98], which is a generalization of the two most popular penalized regression methods, namely, the l_2 penalty of the ridge regression [HK70], and the l_1 penalty of the least absolute shrinkage and selection operator (Lasso) [Tib96]. In the Bayesian shrinkage approach, a prior distribution $p(\beta|\lambda)$ is assigned to β. The posterior distribution given the observed data takes the form of

$$p(\beta|y, \lambda) \propto p(y|\beta)p(\beta|\lambda). \tag{1.7}$$

If we are looking for β that maximizes the posterior distribution, the problem is equivalent to finding

$$\hat{\beta} = \arg\max_{\beta} \left[\log p(y|\beta) + \sum_{j=1}^{p} \log p(\beta_j|\lambda) \right] \tag{1.8}$$

which is equivalent to the penalized ML method. Then $\hat{\beta}$ is the mode of the posterior distribution and this method is referred to as *maximum a posteriori* (MAP) approach [GCS+13].

The Bayesian interpretation of penalized regression methods has sparked interest in developing Bayesian hierarchical regression models for model selection. Theoretically, prior distributions with spike finite limit at zero and flat tails at two ends can be a penalty of the log posterior distribution (Figure 1.5) [HWIB08, AC10, HL13]. In practice, conjugate priors are often chosen [GCS+13]. Among them the Normal + *inv*-χ^2 (t-family) distribution [Xu07, GJPS08, YB09, Xu10] and NEG hierarchical priors [HWIB08, GB11] are most often used with both variable shrinkage and computational feasibility considerations.

For Bayesian estimation, Markov Chain Monte Carlo (MCMC) [RC04] can be employed to draw samples from posterior distribution for each parameter. However, for high dimensional data the MCMC method is known to be computationally intensive. The posterior mode estimation is a modal representation of the posterior distribution that achieves faster computation and easier interpretation. Efficient methods for estimating the posterior mode have been developed, which employ the expectation-maximization (EM) algorithm [Gel06, GJPS08, YB09, Xu10] or other numerical methods that further integrate out the variance components and obtain similar sparse results as the Lasso (which is named as HyperLasso) [HWIB08, AC10].

1.3.5 Motivation and Objectives for QTLs Mapping

The l_2 penalty in ridge regression cannot shrink regression coefficients to zero even though it guarantees a solution for $p \gg n$ problems. The l_1 penalty in the Lasso and the Bayesian shrinkage in the HyperLasso are able to set the coefficients to zero

and estimate only nonzero variables, thus offering computational feasibility to handle a large number of variables [HWIB08, WCH$^+$09]. However, Lasso and HyperLasso only provide a point estimate of the variables without other information, such as the variance of the estimated effects [Tib96, HWIB08, AC10].

The EM algorithm in inferring Bayesian hierarchical models makes multiple passes over all candidate variables and produces a sequence of MAP estimates. Given the slow convergence problem [GJPS08, YB09, Xu10, GB11], it is computationally expensive when dealing with high dimensional data. Even though the Bayesian shrinkage approach used in [CHX11] has shown improved performance, the greedy coordinate ascent algorithm makes it slower when dealing with a larger number of parameters.

To address this, we develop fast and robust sparse model learning techniques and design a powerful hypothesis test method for the QTL mapping problem. We are particularly interested in the Bayesian shrinkage approach with appropriate prior distribution for the unknown variables that not only shrinks irrelevant variables, yielding a sparse model, but also provides both posterior mode and covariance estimates to facilitate hypothesis tests. Furthermore, we employ a proximal gradient method that considers the nonzero effects in the model simultaneously over the greedy coordinate ascent approach taken in [CHX11]. Such designs result in sparse models with extremely fast convergence, which is ideal for analyzing high dimensional genotype data efficiently and more accurately.

1.4 Outline of the Dissertation

The rest of this dissertation is organized as follows. In Chapter 2, we describe our proposed MRF framework for modeling TFBSs and develop two ML based methods to estimate model parameters using L_1 and Bayesian regularization, respectively. Furthermore, using the feature induction approach in [PPL97] and our parameter estimation method, we develop an iterative algorithm that can find a near-optimal MRF model for the TFBSs. In Chapter 3, to further reduce the complexity of the algorithm to deal with long motifs, we develop a pairwise MRF model for TFBSs. Based on this pair-wise MRF, we formulate an optimization problem to approximate the partition function that significantly reduces the computational complexity.

In Chapter 4, we present a fast empirical Bayesian Lasso (EBlasso) method in conjunction with proximal gradient algorithm using NE hierarchical prior distribution for multiple QTL mapping [AHC14]. In Chapter 5, we further extend this model with NEG hierarchical prior distribution with fast proximal gradient algorithm to reduce the computational complexity while increasing the detection rate. Furthermore, we develop a powerful hybrid model that is capable of detecting more QTL effects with lower false positives.

Although the sparse model learning work presented in this thesis is used in the context of TFBSs identification or QTL mapping, the algorithms are equally applicable to a broad range of problems, such as *de novo* TFBSs identification, whole-genome QTL mapping and pathway-based genome-wide association study (GWAS) [HXC14b,

HMVC14], etc. The final chapter concludes the dissertation by summarizing the achievements and opening up avenues for future research.

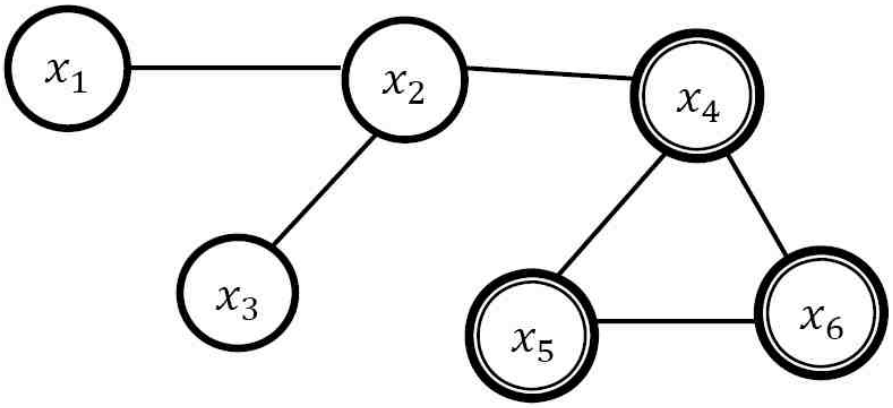

Figure 1.4: An undirected graph

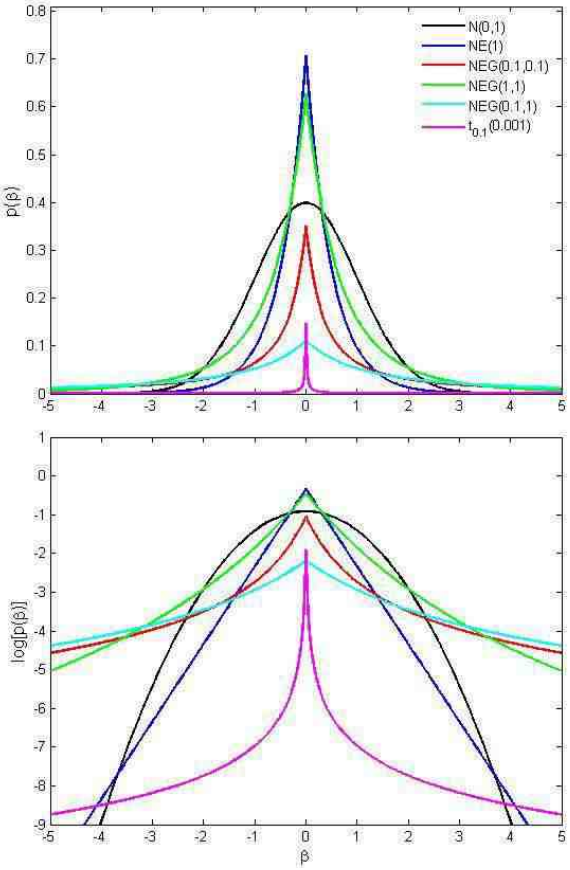

Figure 1.5: Prior distribution that penalizes posterior distribution. Top: prior probability of regression coefficients; bottom: log scale of the prior probability of regression coefficients.

CHAPTER 2

Modeling TFBSs Identification Problem with MRF

We present a framework for identifying TFBSs with MRF. For a given DNA sequence, the segments corresponding to TFBSs are known as foreground, while the rest are considered as the background. We first briefly introduce the background models that are widely used in the research community. Then we formulate the TFBSs identification as an undirected graphical model structure learning problem. We further define a set of features that can be used with the structure learning problem using the factor graph concepts. Using these features in the model allows us to convert the problem into a parameter estimation problem.

A fully connected structure leads to a computational explosion that makes the model learning intractable. To address this, we propose a two-level feature reduction technique that will be described in sections 2.3.2, 2.3.3, and 2.3.4. This reduces the complexity while capturing dependencies among nucleotides. The efficient implementation of the algorithm also plays a vital role, and we will discuss some of the important aspects of the algorithms implementation in section 2.4.

2.1 The Nature of the Problem

We consider identification of TFBSs as a classification problem that classifies a given segment of a DNA sequence into a TFBS or a background sequence (non-TFBS sequence). Suppose we have a set of experimentally discovered TFBSs for a specific transcription factor. We use this set of sequences as training data to learn the parameters of the model for TFBSs or the foreground model. Suppose that we also have a set of background DNA sequences for non-TFBS sequences. We use this set of sequences to train the background model (see section 2.2 for more details). Specifically, we use MRF based foreground models and employ a widely used Markov model for background sequences. Based on these two models, we design a classification rule to maximize the true positive (TP) rate for a fixed true negative (TN) rate. Before going into more details of our foreground model, let us briefly summarize the widely used background models.

2.2 Background Models

Various types of background models have been proposed for modeling non-TFBS regions of the genome. However, the models based on Markov properties have drawn great interest in this area [HETC00]. Usually, the background sequences are of variable lengths. Therefore, all background models can be considered as homogeneous models where there is no position-dependent information.

In general, Markov models can be either fixed order or variable order, but the most popular background model is the zeroth order Markov model, also known as

Bernoulli model due to its simplicity [LNL95, HETC00]. We abbreviate this model as *Markov(0)*. Let us now define some notations to continue with our discussion.

We denote a segment of a DNA sequence starting from position l, and ending with position n as $x_l^n = x_l, x_{l+1}, \cdots, x_{n-1}, x_n$ [BW99]. With this notation, we can define any DNA sequence of length N as x_1^N. For simplicity, when we use the whole sequence, we drop the superscript and subscript, and use x to denote the sequence. Let each position of this sequence x_i take values from a configuration set denoted by $\mathcal{A} = \{A, T, G, C\}$, which is also known as nucleotide bases or simply DNA letters [BW99]. Clearly, the cardinality of this set, d, is equal to 4. Furthermore, we define each element in $\mathcal{A}$ as a_j where $j = \{1, 2, 3, 4\}$. Thus we have $a_1 = A$, $a_2 = T$, $a_3 = G$ and $a_4 = C$. Let X denote a random DNA sequence and X_i be the random variable associated with the i^{th} position of X so that X_i takes values from the same configuration set $\mathcal{A}$. With these notations, for a zeroth order homogeneous background model, we can write the probability of a sequence x, denoted by $p(x) \equiv p(X = x)$ as

$$p(x) = \prod_{i=1}^{N} P(X_i = x_i). \tag{2.1}$$

Although this zeroth order model is very simple, it has no context (the set of nucleotides observed earlier is called the context) information at all. Therefore, this model poorly captures the complex dependencies residing in DNA sequences [TLM+01, LBL01]. For this reason, higher order Markov models have been

proposed. With this representation, the likelihood of a sequence given by an L^{th} order Markov model, $Markov(L)$, can be written as

$$p(x) = \prod_{i=1}^{N} P(X_i = x_i \mid X_{i-L}^{i-1} = x_{i-L}^{i-1}), \tag{2.2}$$

where $i-L \equiv max(i-L, 1)$. As we can see from (2.2), the likelihood is based on the sequence of predecessors of a fixed length L, where $L < N$. For example, the likelihood of the sequence ACCTGGCT, $P(ACCTGGCT)$, with a second-order Markov model can be written as $P(A)P(C|A)P(C|AC)P(T|CC)P(G|CT)P(G|TG)P(C|GG)P(T|GC)$. The main drawback of using fixed-order Markov models is the exponential growth of parameters with respect to the order of the model. This will result in over-fitted models as indicated by Buhlmann et al. [BW99] and Orlov et al. [OFPK02]. To alleviate this problem, variable order Markov models (VOM) were proposed.

Both fixed order and variable order Markov models can be represented as trees, usually known as context trees [Ris83]. The height of a tree is equal to the order of the Markov model, while the number of leaves is equal to d^L. Each node contains d number of probabilities corresponding to each nucleotide base. Therefore, the zeroth order Markov model has just a single node at the root. Figure 2.1 represents a first-order Markov model as a context tree. The branches from the root on top down to the leaves represent the reversed context.

The VOM tree is constructed from a fixed order Markov model as described in [BGMS03, BW99]. To construct the VOM tree from a regular fixed order Markov model, the concept of pruning is used. The statistical significance of the parent

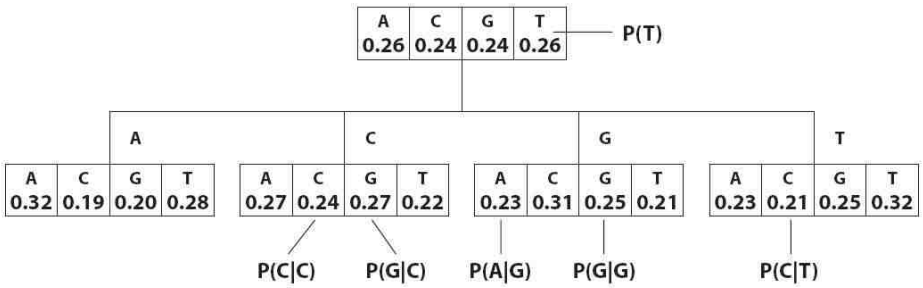

Figure 2.1: A first order Markov tree, $Markov(1)$

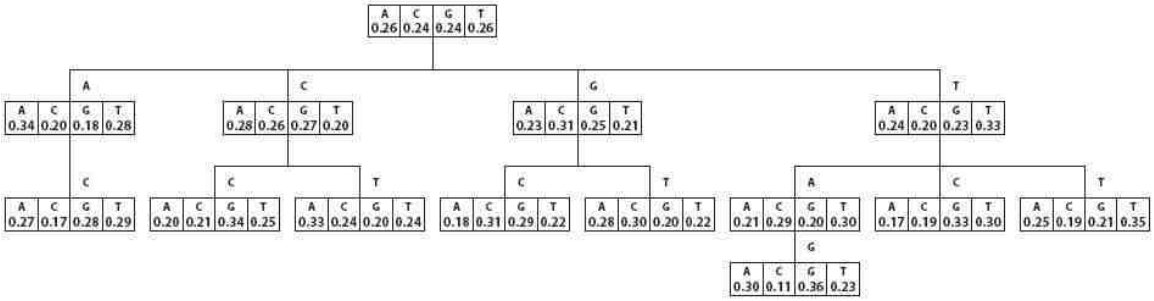

Figure 2.2: A Homogeneous VOM tree constructed from a fifth-order Markov model.

node and child node of the tree is measured using KL divergence. To obtain the final VOM tree, we recursively prune out all child nodes having smaller KL divergence (usually known as pruning constant, c) than their corresponding parent. The number of nodes contained in the final tree is governed by the value of the pruning constant. The higher the value of the pruning constant, the smaller the number of nodes we get in the final tree. For further details of constructing VOM trees, see [SBG07, BGSG$^+$05, BGMS03, BW99]. Figure 2.2 shows a VOM tree constructed from a fifth-order Markov model. It can be clearly seen that most of the nodes are pruned out and the height of the tree has become 3 instead of 5. Thus, it eliminates most of the less relevant dependencies, which helps overcome the problem of over-fitting.

The likelihood of a sequence using a VOM model can be written in the same way as fixed order Markov models with slight modification to the order L as [BGSG$^+$05],

$$p(x) = \prod_{i=1}^{N} P(X_i = x_i \mid X_{i-L_i}^{i-1} = x_{i-L_i}^{i-1}). \tag{2.3}$$

Here the number of previously observed symbols to calculate the probability of each position is not fixed as we saw in fixed order Markov models. The optimum value of L_i is given by, $L_i \equiv max\{L - j | x_{L-j} \in \mathcal{B}\}$, where j is an integer that can take values $\{0, \cdots, L\}$ and the set $\mathcal{B}$ is the set containing all predecessors of x_i remaining in the VOM tree. If $(L - j) = 0$ then there is no context and we use the corresponding probability from the root. For example, when calculating probability for the position i, if we do not find the context $x_{i-L}, x_{i-L+1}, \cdots, x_{i-1}$ in the VOM tree, we look for the context $x_{i-L+1}, x_{i-L+2}, \cdots, x_{i-1}$. This process is continued until we find an entry in the tree. Moving back to Figure 2.2, it is not difficult to show that the calculation of likelihood of the sequence TCCGGA, P(TCCGGA) is reduced to $P(T)P(C|T)P(C|TC)P(G|CC)P(G|CG)P(A|G)$.

Proof:

$$\begin{aligned}
P(TCCGGA) &= P(T)P(C|T)P(C|TC)P(G|TCC)P(G|TCCG)P(A|TCCGG) \\
&= P(T)P(C|T)P(C|TC)P(G|CC)P(G|CCG)P(A|CCGG) \\
&= P(T)P(C|T)P(C|TC)P(G|CC)P(G|CG)P(A|CGG) \\
&= P(T)P(C|T)P(C|TC)P(G|CC)P(G|CG)P(A|GG) \\
&= P(T)P(C|T)P(C|TC)P(G|CC)P(G|CG)P(A|G)
\end{aligned}$$

■

In each step we remove the element x_{i-L} from the context when we do not find the entry in the tree. As an example, the last term of the likelihood, $P(A|TCCGG)$, is all the way reduced to $P(A|G)$ since we do not find any entry in the tree for $TCCGG$, $CCGG$, CGG and GG when A is observed. With this brief understanding of the widely used background models, let us now present our MRF framework for the foreground model.

2.3 MRF Model for TFBSs

An MRF is an undirected graph $G = (V, E)$, where $V = \{1, \cdots, N\}$ is the vertex set and E is a set of undirected edges joining pairs of vertices. To model a DNA sequence as an MRF, we define each position in the sequence as a vertex or node. To each node $i \in V$, we associate a random variable X_i which in our case takes values in the configuration space $\mathcal{A}$. A clique of the MRF is a maximum subset of V in which each pair of nodes is joined by an edge. All random variables associated with nodes in a clique are dependent on each other. The joint distribution of a random sequence X, $p(x) \equiv p(X = x)$, associated with an MRF, can be written as a product of potential functions $\psi(X_C)$ over the maximal cliques of the graph as [Bis06]

$$p(x) = \frac{1}{Z} \prod_C \psi(X_C), \tag{2.4}$$

where C represents a clique, and X_C denotes the connected nodes in clique C. The product is taken over all possible cliques in the graph. As explained in Section 1.2.5,

we need to use an external normalization factor called partition function,

$$Z = \sum_{\mathcal{X}} \prod_{C} \psi(X_C), \tag{2.5}$$

If not normalized, the product of potential functions cannot be interpreted as a probability distribution. Here the summation is taken over all possible sequences of length N denoted by $\mathcal{X}$. Let us denote $f_s(X_S)$ as a feature function over a subset of nodes X_S within a clique, which we will define shortly. Since we only need to consider potential functions as strictly positive, each potential function can be written as an exponential function of a set of feature functions [KFL01]

$$\psi(X_C) = \exp\Big(\sum_{s} \theta_s f_s(X_S)\Big), \tag{2.6}$$

where θ_s is the weight for the feature function $f_s(X_S)$. Using this representation, we can rewrite the joint distribution in (2.4) as

$$p(x) = \frac{1}{Z} \exp\Big(\sum_{k=1}^{K} \theta_k f_k(X_k)\Big), \tag{2.7}$$

where K is the total number of feature functions. For this joint distribution, we obtain the partition function as

$$Z = \sum_{\mathcal{X}} \exp\Big(\sum_{k=1}^{K} \theta_k f_k(X_k)\Big). \tag{2.8}$$

Also, we denote the set containing the weights of all feature functions as Θ $(\forall k, \theta_k \in \Theta)$. As one can see from (2.8), Z is a function of Θ. Therefore, we will alternatively use $Z(\Theta)$ to denote the partition function Z, whenever we need to emphasize that the Z is a function of Θ.

To complete the model, we need to define all feature functions. We define a set of $4N$ elementary feature functions known as atomic or first-order feature functions as follows:

$$f_{4(i-1)+j} = \begin{cases} 1, \text{ if } x_i = a_j \\ \\ 0, \text{ otherwise,} \end{cases} \tag{2.9}$$

where $i = 1, \cdots, N$ and $j = 1, \cdots, 4$, $a_1 = A$, $a_2 = T$, $a_3 = G$ and $a_4 = C$. If we denote this set of feature functions as f_n, where $n = 1, \cdots, 4N$, then we can find the correlation between n, i and j as $i = \left\lceil \frac{n}{4} \right\rceil$ and $j = n\%4 + 1$. To further clarify, $f_1 = 1$ if $x_1 = A$, $f_6 = 1$ if $x_2 = G$ and so forth. We denote this set of atomic feature functions as $\mathcal{F}_{\text{atom}}$. The motivation behind this definition is that we can define any higher order feature functions as a product of these elementary feature functions, as we will explain shortly. Now let us define a second-order feature function as:

$$f_m = \begin{cases} 1, \text{ if } x_{i_1} = a_{j_1} \text{ AND } x_{i_2} = a_{j_2} \\ \\ 0, \text{ otherwise,} \end{cases} \tag{2.10}$$

where $m = 4N + 1, \cdots, 8N(N + 1)$, $i_1 = 1, \cdots, N - 1$, $i_2 = i_1 + 1, \cdots, N - 1$, $j_1 = 1, \cdots, 4$ and $j_2 = 1, \cdots, 4$. Capitalizing on the idea of atomic features, we can find an alternative form given by

$$f_n = f_{4(i_1-1)+j_1} \times f_{4(i_2-1)+j_2}. \tag{2.11}$$

As one can see, the second-order feature function can be defined as the product of two atomic features. This process can be continued to obtain any higher order feature functions. In general, a k^{th} $(k > 1)$ order feature function can be defined as a product of k atomic feature functions.

If the MRF model contains only the first-order feature functions, then all nodes are independent and the model is equivalent to the PWM model. If the MRF model contains all first and second order feature functions, then it is the pairwise MRF considered in [WJ06]. Given the probability model specified by the probability distribution (2.7), we need to estimate model parameters $\theta_k,\ k = 1, \cdots, K$, which we will discuss in the next section.

2.3.1 Parameter Estimation

Suppose that we are given a set of M independent and identically distributed training sequences $D = \{x^j, ..., x^M\}$. Then the log likelihood function of the data can be written as

$$l(D|\Theta) = \sum_{k=1}^{K} \theta_k f_k(D) - M \log Z(\Theta), \tag{2.12}$$

where $\Theta = \{\theta_1, \cdots, \theta_K\}$, we have explicitly expressed Z as a function of Θ, and

$$f_k(D) = \sum_{m=1}^{M} f_k(x^m) \tag{2.13}$$

is the sum of the feature values over the entire data set. We can estimate the set of parameters, Θ, by maximizing the log likelihood function (2.12). It can be easily shown that this log likelihood function is convex. Therefore, maximization (2.12) becomes a convex optimization problem that can be solved by using any convex optimization algorithm.

We allow feature functions of any possible order (up to Nth order) to be included in our model to capture dependencies among several nodes. Note that there are $4^k\binom{N}{k}$ k^{th} order feature functions for motifs of length N. If we include higher order feature

functions, then the model contains a large number of parameters even for a moderate N. This exponential number of features causes very high computational complexity in the ML approach. Furthermore, the limited number of training instances leads to over-fitted models.

To tackle this challenge, we propose to use an iterative method to build the model by carefully introducing only the most relevant set of features in each iteration by using a feature selection algorithm [PPL97]. Specifically, we start the graph structure with a small set of relevant nodes. And then, in each iteration, we introduce the next set of the most relevant nodes and edges to the current model, thus adding more and more dependencies among nodes.

This mainly limits the number of features associated with the ML estimation process. Then we apply another level of feature reduction technique by using a regularization together with the ML approach. In fact, we propose to use two types of regularization: L_1 *regularization* [SK03, Ng04, DPS04, LGK07] and *Bayesian Regularization* [CT06]. This will eliminate most of the excessive features involved in the current model, which helps to reduce the number of features added by the feature induction algorithm in the next iteration.

These two techniques drastically reduce the computation complexity while increasing the performance, as we will shortly see in the simulation and results section (Section 2.6). Now, we describe each of these method in detail in the next two sections.

2.3.2 L_1 Regularization

In this approach, we first get a L_1-regularized likelihood function by inserting a term of weighted L_1 norm of the parameters into (2.12) and then find parameters by maximizing this L_1-regularized likelihood function:

$$\arg\max_{\Theta} \left[\sum_{k=1}^{K} \theta_k f_k(D) - M \log Z(\Theta) - \lambda \sum_{k=1}^{K} |\theta_k| \right], \tag{2.14}$$

where λ is a constant. We will later discuss determining the value of λ. Using L_1-regularization provides several advantages. First, it has been shown that L_1-regularization leads to a sparse model by nullifying less relevant parameters [Tib94, LGK07], which in turn can avoid the over-fitting problem. Second, it can identify relevant features with fewer number of training samples [DPS04, Ng04]. These two advantages ensure that the L_1-regularized ML estimation offers much better performance than the direct ML estimation.

Since both the generalization ability of the classifier and the level of sparsity achieved are critically dependent on the value of the regularization parameter λ, it must be carefully tuned to optimize performance. Higher values of λ keeps only the more sensitive features while lower values of λ includes less sensitive features. Therefore, selecting a suitable value of λ depends on the training data. At the same time, using a fixed value for λ does not give good results. Therefore, we use the variable λ selection method with the aid of simulated annealing schedule to address the problem. This allows us to add more sensitive features at the first and less and less sensitive features to be included later in training process.

2.3.3 Bayesian Regularization

The second regularization we use is Bayesian regularization that was originally proposed to solve sparse logistic regression problems [CT06]. The main advantage of this method is that it does not contain any parameter involved with the regularization term as in L_1 norm. The regularization parameter λ is integrated out analytically, using an uninformative Jeffery's prior, in the style of Buntine and Weigend [CT06, BW91]. This provides an easy way of using this regularization together with the ML approach, since we do not need to employ other complicated methodology to optimize the regularization parameter as we suggested with L_1 norm. Incorporating this regularization term into (2.12), we get the Bayesian regularized ML optimization function:

$$\arg\max_{\Theta} \left[\sum_{k=1}^{K} \theta_k f_k(D) - M \log Z(\Theta) - K \log \left(\sum_{k=1}^{K} |\theta_k| \right) \right]. \tag{2.15}$$

With L_1 regularization, we need to use a separate data set known as *evaluation* set to find the best model, as explained in Section 2.4.3. This will effectively reduce the number of data instances for the training process. But we can overcome this problem with Bayesian regularization.

To give some intuition to this fact, consider the first two terms of the (2.15). They come from the original likelihood function. When the number of features is increased, the net value of these two terms also increases because the model captures the less sensitive information from the training set. This is the reason for the over-fitting problem. The regularization term is always positive and it is directly proportional to the number of features, as can be seen from (2.15). When we get more and more

features, the incremental value of this term becomes larger. Since this is subtracted from the net value of the first two terms, when the number of features in the model is greater than a certain value, the overall maximum value of the (2.15) becomes smaller. Therefore, the maximum likelihood values of (2.15) as a function of model index follow a convex shape. This can be clearly seen with our results presented in Section (2.6). We use this fact to eliminate the need for another evaluation set for the model selection process, as described in Section (2.4.3).

2.3.4 Feature Selection Algorithm

Clearly, there is no closed-form solution to the optimization problem in (2.14) or (2.15). Therefore, we need to employ numerical optimization algorithms to find the solution. The main challenge with the application of an optimization algorithm to this problem is the calculation of the partition function $Z(\Theta)$. With the length of N base pairs for the TFBS, we need to get a sum over $4^N \times K$ terms to evaluate $Z(\Theta)$, which requires very large computation if both N and K are relatively large.

Therefore, it is very important that, for a given N, we choose a relatively small K by only selecting the most relevant set of features before carrying out the optimization step. Even though we have discussed two types of regularization techniques capable of eliminating most of the excessive features, they do not have any control over the number of features (K) included in the initial ML estimation problem. This is the main motivation to use a feature induction algorithm.

The basic strategy of this method is to add a new set of features to the current model so that the overall model gets closer to a reference distribution $\bar{p}$, which is the empirical distribution of the training set defined as

$$\bar{p}(x) = \frac{c(x)}{M}, \tag{2.16}$$

where M is the total number of training sequences and $c(x)$ is the count of occurrences of sequence x.

Assume that the probability distribution of the sequence x given by the current model is $q_i(x)$. Given this distribution, our goal is to find the next best set of features to include in the model. We start with a set of active features determined in the current model, say $\mathcal{F}_{\text{cur}}$. As discussed in Section 2.3, any higher order feature functions can be obtained as a product of several atomic feature functions. Therefore, we select one feature at a time from $\mathcal{F}_{\text{atom}}$ and multiply it with each active feature in $\mathcal{F}_{\text{cur}}$ to build the next candidate features. If the selected atomic feature is not included in the current model, the atomic feature itself is also added as a candidate feature. Now, we calculate the gain G of adding each candidate feature to the current model as

$$G = D_{KL}(\bar{p}\|q_i) - D_{KL}(\bar{p}\|q_{i+1}), \tag{2.17}$$

where q_{i+1} is the probability distribution of x obtained from the new model after adding the candidate feature.

To reduce computational complexity, we assume that the existing feature weights remain unchanged while a new feature is added as in [PPL97]. Note that the weight of the newly added feature can be optimized to calculate the maximum gain. Let $\hat{\theta}$ be the weight of the newly added feature that gives the maximum gain. With the

assumption of fixed weights with current model feature functions, and using indicator features defined in Section 2.3, we can get a closed-form solution for the new feature weight $\hat{\theta}$ as:

$$\hat{\theta} = \log\left(\frac{\bar{p}[g](1-q_i[g])}{q_i[g](1-\bar{p}[g])}\right), \tag{2.18}$$

where $p[g] = \sum_{x\in\mathcal{X}} g(x)p(x)$ and $g(x)$ denotes the added feature function. With this optimum weight of the new feature function, the gain can be written as [PPL97]

$$G = (1-\bar{p}[g])\log\left(\frac{1-\bar{p}[g]}{1-q_i[g]}\right) + \bar{p}[g]\log\left(\frac{\bar{p}[g]}{q_i[g]}\right). \tag{2.19}$$

Proof: By applying the definition of KL divergence to (2.17),

$$\begin{aligned} G &= D_{KL}(\bar{p}\|q_i) - D_{KL}(\bar{p}\|q_{i+1}) \\ &= \sum \bar{p}(x)\log\left(\frac{\bar{p}(x)}{q_i(x)}\right) - \sum \bar{p}(x)\log\left(\frac{\bar{p}(x)}{q_{i+1}(x)}\right) \\ &= \sum \bar{p}(x)\log\left(\frac{q_{i+1}(x)}{q_i(x)}\right) \\ &= \sum \bar{p}(x)\log\left(\frac{\exp\left(\sum \theta_k f_k\right)\exp\left(\alpha g\right)}{q_i(x)\sum \exp\left(\theta_k f_k\right)\exp\left(\alpha g\right)}\right) \\ &= \sum \bar{p}(x)\log\left(\frac{\exp\left(\alpha g\right)}{q_i[\exp\left(\alpha g\right)]}\right) \\ &= \sum \bar{p}(x)(\alpha g - \log q_i[\exp\left(\alpha g\right)]) \\ &= \alpha\bar{p}[g] - \log q_i[\exp\left(\alpha g\right)] \end{aligned} \tag{2.20}$$

The derivative of (2.20) with respect to α yields

$$\frac{\partial G}{\partial \alpha} = \bar{p}[g] - \frac{q_i[g\exp\left(\alpha g\right)]}{q_i[\exp\left(\alpha g\right)]}.$$

Since we consider only binary features, g can take only 1 or 0. Let X_0 be the set of x that evaluates to 0 and X_1 be the set of x that evaluates to 1. Now at the maximum of the above equation, the derivative is equal to 0.

$$
\begin{aligned}
\bar{p}[g] - \frac{q_i[g \exp(\hat{\alpha} g)]}{q_i[\exp(\hat{\alpha} g)]} &= 0 \\
\bar{p}[g] &= \frac{\sum \exp(\theta_k f_k) g \exp(\hat{\alpha} g)}{\sum \exp(\theta_k f_k) \exp(\hat{\alpha} g)} \\
&= \frac{\sum_{X_1} \exp(\theta_k f_k) \exp \hat{\alpha}}{\sum_{X_0} \exp(\theta_k f_k) + \sum_{X_1} \exp(\theta_k f_k) \exp \hat{\alpha}} \\
\exp \hat{\alpha} &= \frac{\bar{p}[g] \sum_{X_1} \exp(\theta_k f_k)}{(1 - \bar{p}[g]) \sum_{X_0} \exp(\theta_k f_k)} \\
&= \frac{\bar{p}[g](1 - q_i[g])}{(1 - \bar{p}[g]) q_i[g]} \\
\hat{\alpha} &= \log \left(\frac{\bar{p}[g](1 - q_i[g])}{(1 - \bar{p}[g]) q_i[g]} \right)
\end{aligned}
$$

Now substituting the $\hat{\alpha}$ into (2.20), the maximum gain provided by the added new feature can be written as

$$
\begin{aligned}
G_{max} &= \hat{\alpha} \bar{p}[g] - \log q_i[\exp(\hat{\alpha} g)] \\
&= \hat{\alpha} \bar{p}[g] - \log (\exp \hat{\alpha} q_i[g] + (1 - q_i[g])) \\
&= \bar{p}[g] \log \left(\frac{\bar{p}[g](1 - q_i[g])}{(1 - \bar{p}[g]) q_i[g]} \right) - \log \left(\frac{1 - q_i[g]}{1 - \bar{p}[g]} \right) \\
&= \log \left(\frac{\bar{p}[g]^{\bar{p}[g]} (1 - q_i[g])^{\bar{p}[g]-1}}{q_i^{\bar{p}[g]} (1 - \bar{p}[g])^{\bar{p}[g]-1}} \right) \\
&= \bar{p}[g] \log (\bar{p}[g] + (\bar{p}[g] - 1) \log(1 - q_i[g])) \\
&\quad - \bar{p}[g] \log q_i[g] - (\bar{p}[g] - 1) \log(1 - \bar{p}[g]) \\
&= (1 - \bar{p}[g]) \log \left(\frac{1 - \bar{p}[g]}{1 - q_i[g]} \right) + \bar{p}[g] \log \left(\frac{\bar{p}[g]}{q_i[g]} \right).
\end{aligned}
$$

■

After calculating gains for all possible candidate features, we add those features with the maximum gain as the next best features to the current model. After adding

Table 2.1: Feature selection algorithm

FeatureSelection
  for all $f_a \in \mathcal{F}_{\text{atom}}$ **do**
    for all $f_c \in \mathcal{F}_{\text{cur}}$ **do**
      $f_n = f_a.f_c$
      if $f_n \in \mathcal{F}_{\text{cur}}$ **then**
        Continue
      end if
      Compute Gain for f_n(2.19)
      Store Gain and f_n
    end for
    if $f_a \notin \mathcal{F}_{\text{cur}}$ **then**
      $f_n = f_a$
      Compute Gain for f_n(2.19)
      Store Gain and f_n
    end if
  end for
  $\mathcal{F}_{\text{max}} = f_n$s with maximum gain
  $\mathcal{F}_{\text{cur}} = \mathcal{F}_{\text{cur}} \cup \mathcal{F}_{\text{max}}$
EndFeatureSelection

those features, we need to use the numerical optimization technique we explained earlier to find the correct feature weight that maximizes the ML optimization function. The feature selection algorithm can be summarized as in Table 2.1. So far we have described two levels of feature reduction techniques we used in our framework. Now we combine these two techniques and formulate the final algorithm.

2.4 Algorithm

We propose to use an iterative method to learn the MRF model. In the initialization phase, we select one atomic feature at a time from $\mathcal{F}_{\text{atom}}$ to form the model. Note that there is only a single weight involved in the model denoted as θ_k. Then we find the optimum value for each θ_k as $\hat{\theta}_k$ using the regularized ML method described earlier.

Since we have only a single feature in the model, the computational overhead can be reduced by deriving a closed-form solution for $\hat{\theta}_k$ when optimizing with (2.14). The derivation is as follows.

There is only a single weight involved in the model, denoted as $\hat{\theta}_k$. Consider the simplified partition function for the model having only a single feature,

$$\begin{aligned} Z &= \sum_{\mathcal{X}} \exp\Big(\theta_k f_k(X_k)\Big) \\ &= 4^{n-1} \exp\theta_k + (4^n - 4^{n-1}). \end{aligned} \tag{2.21}$$

With this Z, we can get the derivative of (2.14) and equate it to zero to find the solution for $\hat{\theta_k}$. Since the derivative is not defined at zero, we can find two solutions for $\hat{\theta_k}$ as follows:

$$\hat{\theta}_k = \begin{cases} \log\left(\frac{-3(f_k(D)-\lambda)}{f_k(D)-M-\lambda}\right), \text{ for } \hat{\theta}_k > 0; \\ \log\left(\frac{-3(f_k(D)+\lambda)}{f_k(D)-M-\lambda}\right), \text{ for } \hat{\theta}_k < 0. \end{cases} \tag{2.22}$$

We find the solution from each case and check whether it matches with the correct range. Due to the convexity of our problem, a unique solution exists for $\hat{\theta}_k$. Therefore, if both cases fail, we can safely assume that the solution to $\hat{\theta}_k$ is 0.

We denote the probability distribution given by each model in the initialization process as $q_0(x)$. Clearly, there are $4N$ number of models to consider. Then we find the KL divergence between $\bar{p}(x)$ and $q_0(x)$ as

$$D_{KL}(\bar{p}\|q_0) = \sum_{x\in\mathcal{X}} \bar{p}(x) \log\left(\frac{\bar{p}(x)}{q_0(x)}\right). \tag{2.23}$$

Since there are $4N$ atomic features, we get $4N$ number of KL distances. The set of atomic features with the minimum KL distance is selected to be included in the

initial model. Based on this set of selected features, we use regularized ML method to evaluate optimum set of Θ, as $\hat{\Theta}$ and form the initial model $q_1(x)$.

Due to the regularization parameter in the optimization, the final algorithm takes different forms based on the regularization technique we use. Therefore, we present our complete algorithm with each regularization method separately.

2.4.1 Algorithm with L_1 Regularization

We use a simulated annealing schedule to estimate the regularization parameter λ in each iteration. As one can see in the algorithm (Table 2.2), the initialization step automatically selects an appropriate starting value for λ based on the given training set. Therefore, one does not need to have any prior knowledge to select the value of λ. Also, we add not only the minimum KL distance features, but also some features close to the minimum distance, in the initialization step. Similarly, in each iteration, we add some features that are close to the maximum gain in addition to the features with maximum gain. This process increases the performance and reduces the time complexity in the training process, as shown in the results.

As can be seen from the algorithm, we always use the simulated annealing schedule for λ and redo the optimization whenever we fail to get a nonzero weight for any of the newly added features or when the new model has a negative gain. But this is not necessary with Bayesian regularization since no regularization parameter is involved. The complete algorithm is shown in Table 2.2.

Table 2.2: Algorithm with L_1 regularization

Begin
$define\ ZERO = 1.0E-5$
$\lambda = 1000$ (use a large number)
define $\lambda_{frac} = 0.85$
Initialization
repeat
for all $f_a \in \mathcal{F}_{\text{atom}}$ **do**
Form the model with f_a
Compute $\hat{\theta}_k$ (2.22)
if $|\hat{\theta}_k| > ZERO$ **then**
Compute KL distance (2.23)
end if
end for
if $\forall\ \hat{\theta}_k$, $|\hat{\theta}_k| \leq ZERO$ **then**
$\lambda = \lambda.\lambda_{frac}$
Continue
end if
until $\exists\ \hat{\theta}_k$ s.t. $|\hat{\theta}_k| > ZERO$
$\mathcal{F}_{\text{cur}}$ = all features with non zero $\hat{\theta}_k$
Take this as the initial model and optimize (2.14)
if All $|\hat{\theta}_k|$s $\leq$ ZERO **then**
go to Initialization
end if
EndInitialization

while true **do**
FeatureSelection Call the feature selection algorithm in Table 2.1
if $|\mathcal{F}_{\max}| = 0$ **then**
go to End
end if

Optimization
repeat
Optimize the current model (2.14)
if $\forall\ \hat{\theta}_k\ \in \mathcal{F}_{\max}$, $|\hat{\theta}_k| \leq ZERO$ **then**
$\lambda = \lambda.\lambda_{frac}$
Continue
end if
until $\exists\ \hat{\theta}_k\ \in \mathcal{F}_{\max}$ s.t. $|\hat{\theta}_k| > ZERO$
EndOptimization

Compute Gain, G (2.19)
if $(G < 0)$ **then**
$\lambda = \lambda.\lambda_{frac}$
go to Optimization
end if
Save the Model
end while
End

2.4.2 Algorithm with Bayesian Regularization

Let us first discuss the differences between Bayesian regularization and L_1 regularization. One difference is that we add only the features with minimum KL distance in the initialization process. Furthermore, we only add features with maximum gain. Recall that in the previous method we added some set of features that were close to the maximum gain. The other difference is the feature selection procedure. In the previous method, we could always include the next best set of features into the model by reducing the value of λ. However, this is not possible with this method. Therefore, we modify the feature selection algorithm by incorporating optimization into the feature selection algorithm as shown in Table 2.3. Now with this modified version of feature selection algorithm, it is easy to present the complete algorithm with Bayesian regularization as shown in Table 2.4.

So far we have presented MRF model formation algorithms with both regularization methods. To complete our discussion, let us consider the main challenges in implementation of this algorithm and efficient techniques to reduce the computational complexity. The main challenge is the extremely high computational complexity of the partition function. MATLAB is not capable of handling that level of computational complexity. Therefore, we implemented algorithms using C++ with efficient data structures and coding techniques. To make the evaluation of feature functions faster, we converted all records to a binary format. Each nucleotide of the sequence was represented with a two-bit binary number (A=00, T=01, G=10, C=11) so that a length N sequence is represented by a $2N$ bit stream. For an example, a sequence

Table 2.3: Mofified feature selection algorithm for Bayesian regularization

FeatureSelection
  for all $f_a \in \mathcal{F}_{\text{atom}}$ **do**
    for all $f_c \in \mathcal{F}_{\text{cur}}$ **do**
      $f_n = f_a.f_c$
      if $f_n \in \mathcal{F}_{\text{cur}}$ **then**
        Continue
      end if
      Compute Gain for f_n(2.19)
      Store Gain and f_n
    end for
    if $f_a \notin \mathcal{F}_{\text{cur}}$ **then**
      $f_n = f_a$
      Compute Gain for f_n(2.19)
      Store Gain and f_n
    end if
  end for
  Assume that we stored gains in the array G_{array}
  Sort G_{array} in descending order
  for $i = 1$,$|G_{array}|$ **do**
    $\mathcal{F}_{\text{max}}$ = all features at i^{th} location
    $\mathcal{F}_{\text{cur}} = \mathcal{F}_{\text{cur}} \cup \mathcal{F}_{\text{max}}$
    Optimize the model (2.15)
    if $\exists\ \hat{\theta}_k\ \in \mathcal{F}_{\text{max}}$ s.t. $|\hat{\theta}_k| > ZERO$ **then**
      break
    else
      $\mathcal{F}_{\text{cur}} = \mathcal{F}_{\text{cur}} - \mathcal{F}_{\text{max}}$
      Reset weights to original values
      $|\mathcal{F}_{\text{max}}| = 0$
    end if
  end for
EndFeatureSelection

Table 2.4: Algorithm with Bayesian regularization

Begin
  $define\ ZERO = 1.0E-5$
  Initialization
    for all $f_a \in \mathcal{F}_{\text{atom}}$ **do**
      Form the model with f_a
      Compute $\hat{\theta}_k$ (2.15)
      if $|\hat{\theta}_k| > ZERO$ **then**
        Compute KL distance (2.23)
      end if
    end for
    if $\forall\ \hat{\theta}_k$, $|\hat{\theta}_k| \leq ZERO$ **then**
      go to End
    end if
    $\mathcal{F}_{\text{cur}}$ = all features with non zero $\hat{\theta}_k$
    Take this as the initial model and optimize (2.15)
    if All $|\hat{\theta}_k|$s $\leq$ ZERO **then**
      go to End
    end if
  EndInitialization
  repeat
    FeatureSelection Call the feature selection algorithm in Table 2.3
    if $|\mathcal{F}_{\max}| \neq 0$ **then**
      Save the Model
    end if
  until $|\mathcal{F}_{\max}| = 0$
End

1	2	3	4	5	6	7	8	9	10	11	12
A	T	G	C	C	C	C	C	A	A	T	T
00	01	10	11	11	11	11	11	00	00	01	01

Figure 2.3: Representing a nucleotide sequence as a bit stream

of length 12, ATGCCCCCAATT, can be represented as 000110111111111100000101, as shown in Figure 2.3. And also we represented all atomic feature functions with a value and a mask.

$$f_{4(i-1)+j} = \begin{cases} \text{mask} = \ 4^i - 4^{i-1}, \\ \\ \text{value} = \ 4^{i-1} \times (j-1), \end{cases} \tag{2.24}$$

where $i = 1, \cdots, N$, and $j = 1, 2, 3, 4$. Note that all atomic features defined on the same position of the sequence have the same mask but different values. To clarify this further, the atomic feature f_2 ($x_1 = T$) is represented with value 1 (000000000000000000000001) and mask 3 (000000000000000000000011) with sequences of length 12.

This representation provides an efficient way of building complex higher order features as well as an evaluation of feature functions to be implemented with bitwise operations rather than string comparison. This will drastically improve the performance when implemented with C++. As an example, to evaluate the feature value for a given sequence, we can simply consider the Boolean operation (x & $mask[f]$ == $value[f]$) where $mask[f]$ and $value[f]$ are the mask and value of feature f. Furthermore, we can make a complex higher order feature f_{com} by combining two features f_x and f_y as follows:

$$f_{com} \begin{cases} \text{mask}[f_{com}] = \text{mask}[f_x] \text{ OR mask}[f_x] \\ \\ \text{value}[f_{com}] = \text{value}[f_y] \text{ OR value}[f_y]. \end{cases} \tag{2.25}$$

We use dynamic programming concepts to reduce the complexity of certain sections of the algorithm, and also implement the gradient projection algorithm with variable step sizes to have a fast convergence using C++.

2.4.3 Best Model Selection

Algorithms that we proposed in previous sections generate a series of sparse MRF models throughout the iteration process, by adding or removing features from the current model to the next model. However, the problem of under-fitting or over-fitting cannot be avoided completely. Therefore, selecting the best model out of this series of models is very important.

For Bayesian regularization, as explained in Section 2.3.3, the maximum likelihood as a function of model index always has a global maximum. Therefore, we use this fact to select the best model with Bayesian regularization. As our results will show, the variation of the function around the maximum point is very small. Therefore, we select all models where maximum likelihood is 2.5% closer to the peak value. Out of these models, we choose the model with the lowest test threshold (the test threshold is defined in (2.26)) as the best model. Our results indicate that this method improves the TP rate in classification.

With L_1 regularization method, there is no such property as in Bayesian regularization. Therefore, we use the evaluation data set (the procedure of generating an

evaluation set will be described in the next section) to select the best model. Once the model iterations are completed, we choose the model that gives the maximum TP rate with respect to the evaluation set as the best model. Whenever we get more than one model with the same highest accuracy, we choose the one with the lowest test threshold as the best. The main drawback to this procedure is that this will effectively reduce the number of training instances, as indicated in next section.

2.5 Experiment Procedure

For the training and testing purposes, we used two types of data sets: the foreground and background data sets obtained from Ben-gal et al. [BGSG$^+$05]. The foreground data set consists of 238 *E.coli* sigma-70 binding sites of length 12 base pairs, and the background data set consists of 472 intergenic 'non-promoter' sequences of *E.coli* that gives a total of 77,644 nucleotides.

From each data set (foreground and background) we randomly selected 10% as the testing set and the rest 90% as the training sets. Out of these 90% training data, we randomly selected another 10% as the evaluation set for selecting the best model for L_1-based algorithm as described in Section 2.4.3. We repeated this procedure to randomly generate 1000 data sets to evaluate the performance of each method. This process can be named as the 1000-fold stratified-holdout procedure [BGSG$^+$05]. Since the PWM-based model is the most popular model for TFBSs while the VOBN-based method outperforms other algorithms such as those based on PWM, BN, or the fixed-order Markov model [BGSG$^+$05], we compared the performance of our MRF-based

methods with the PWM and VOBN-based methods. Using the training data in each background data set, we trained the background model M_{bg}. For methods, PWM, VOBN, and MRF with Bayesian regularization ($MRFBayes$), we used training data in each foreground data set to train the foreground model M_{fg}. For MRF with L_1 regularization method ($MRFL1$), we used the training data that excludes the evaluation data set. After training our MRF-based methods, we used the evaluation data set and best model selection criteria as described in section 2.4.3 to select the best model for each data set. After training the foreground and background models, we classified each sequence in the testing data set as a TFBS if the following condition was satisfied:

$$\frac{P(x|M_{\text{fg}})}{P(x|M_{\text{bg}})} \geq T, \tag{2.26}$$

where $P(x|M_{\text{fg}})$ and $P(x|M_{\text{bg}})$ represent the probability of x calculated from the foreground and background models, respectively, and T is a threshold that was determined by a given TN rate.

In the process of background model training, we first created a context tree as described in Section 2.2 and then calculated the probability of each node as follows:

$$P\left(X_i = x_i \mid X_{i-L}^{i-1} = x_{i-L}^{i-1}\right) = \frac{n\left(x_{i-L}^{i}\right)}{n\left(x_{i-L}^{i-1}\right)}, \tag{2.27}$$

where $n(.)$ denotes the frequency of its argument in a training set. To compensate for zero occurrences of certain oligonucleotides, we added a fixed count known as *pseudo-count* [BGSG$^+$05] to all the nodes in the same level of the tree before calculating the probability using (2.27). More precisely, we assumed that the sum of all *pseudo-counts* in a level was 4096 [BGSG$^+$05, HGC95]. For example, a fifth-order Markov model

has a *pseudo-count* of 1024 for each frequency in the root node, and *pseudo-count* of 1 for each frequency in leaf nodes.

We conducted three phases of experiments. In the first phase, we chose fixed order Markov models with several orders as the background model. We fixed the TN rate at 99.9% to have a fair comparison with the results of Ben-gal et al. [BGSG$^+$05], and then compared the mean TP rates of foreground models, namely, PWM, $VOBN(1, 2^{-3.75})$, MRFL1, and MRFBayes. For the VOBN model, we used the pruning constant of $2^{-3.75}$ and the initial order of 1, because these parameters give the best performance for the given data set according to [BGSG$^+$05]. For the notational simplicity, in Section 2.6, we alternatively used VOBN to refer to the actual model $VOBN(1, 2^{-3.75})$. In the second phase, we used a fifth-order VOM model with different pruning constants [BGSG$^+$05] to analyze the effect of VOM background models to the mean TP rate. In the third phase, we used all the background models used in first and second phases of experiments to evaluate mean TP rates of each foreground model with different TN rates. Then we selected the best mean TP rate to obtain receiver operating characteristic (ROC) curves.

2.6 Results and Discussion

Before we move into the classification results, we will justify our claim in the best model selection procedure for MRF with Bayesian regularization. We claimed that the maximum likelihood of (2.15) as a function of model iteration index achieves a global maximum. Since all the data sets follow the same pattern, we randomly

selected four data sets for the demonstration, and calculated the maximum likelihood given by models in each iteration.

Figure 2.4 shows the results for sets 1, 5, 100, and 735. In Figure 2.4 the likelihood function clearly achieves a global maximum. Furthermore, the variation of the likelihood is very small around the peak. This is the reason to pick a set of models around the peak point to obtain the best model as described in Section 2.4.3. We use all models that are 2.5% close to the peak value in selecting the best model. As one can see from Figure 2.4, we do not need to go for higher iterations since those models fall out of the domain considered in best model selection. Thus, this provides an early termination criterion for iterations, and further reduces the time required in the training process.

2.6.1 Phase 1

We used fixed order homogeneous Markov models with order $L = 0, 1, \cdots, 4$, as background models. Fixing the TN rate at 99.9%, we obtained the mean TP rates from 1000 data sets for foreground models. Figure 2.5 plots these mean TP rates. Note that the number of nodes for each background model is shown in brackets under the name of the background model.

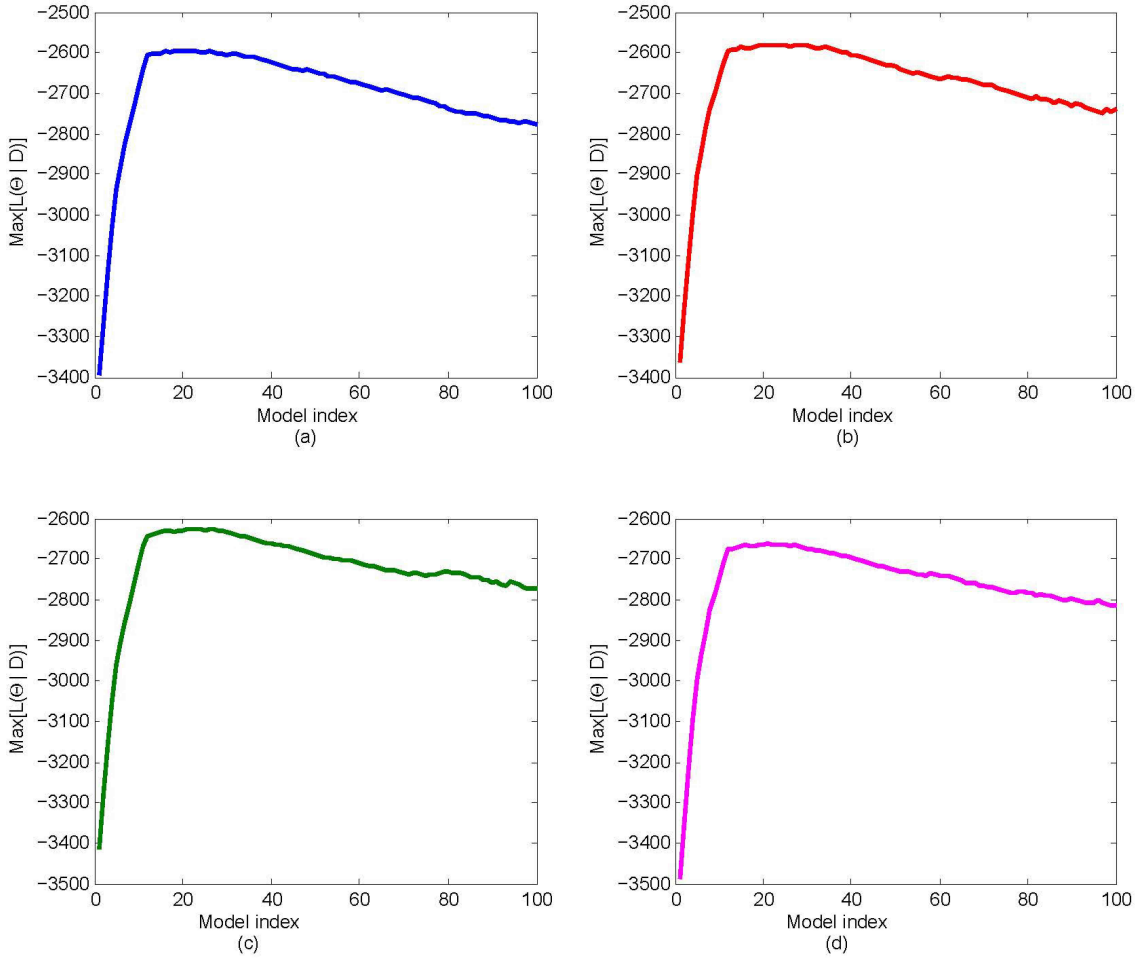

Figure 2.4: Maximum likelihood as a function of model iteration for the foreground model MRFBayes. (a), (b), (c), and (d) represent sets 1, 5, 100, and 735, respectively.

We noticed in the experiments that there was not much improvement when the number of iterations was approximately greater than 100. Therefore, we limited the number of iterations to 100 for each set, and selected the best model as described in Section 2.4.3. Note that there are two additional curves in Figure 2.5, labeled MRFL1-B and MRFBayes-B. To obtain these curves, for each set, we selected the best model by using the testing data itself, and then calculated the mean TP rate given by these best models. Therefore, MRFL1-B and MRFBayes-B can be thought as upper bounds of mean TP rates for MRFL1 and MRFBayes methods, respectively. The reason is that if we could develop an ideal best model selection method, then we could have achieved these mean TP rates.

In Figure 2.5, VOBN and MRFL1 achieve their best TP rates with the third-order Markov background model, while PWM and MRFBayes methods achieve their best TP rate with the second-order Markov background model. Comparing the TP rates for all background models with an order from 0 to 4, we observe that the maximum TP rates for PWM and VOBN are 44.84% and 46.25%, respectively. Clearly, both of our proposed methods outperform the other two methods with all the background models. The maximum TP rate of MRFL1 is 47.42%, while the maximum TP rate of MRFBayes is 48.95%. Thus, our MRFBayes method offers 4.12% improvement over the PWM and 2.71% improvement to the VOBN. Furthermore, it has 1.53% improvement over the MRFL1 method. Using the t-test to test the null hypothesis that the TP rate of our MRFBayes is equal to that of PWM or VOBN, we obtained a *p-value* of 1.34×10^{-15} and 1.21×10^{-7} for the test against PWM and VOBN,

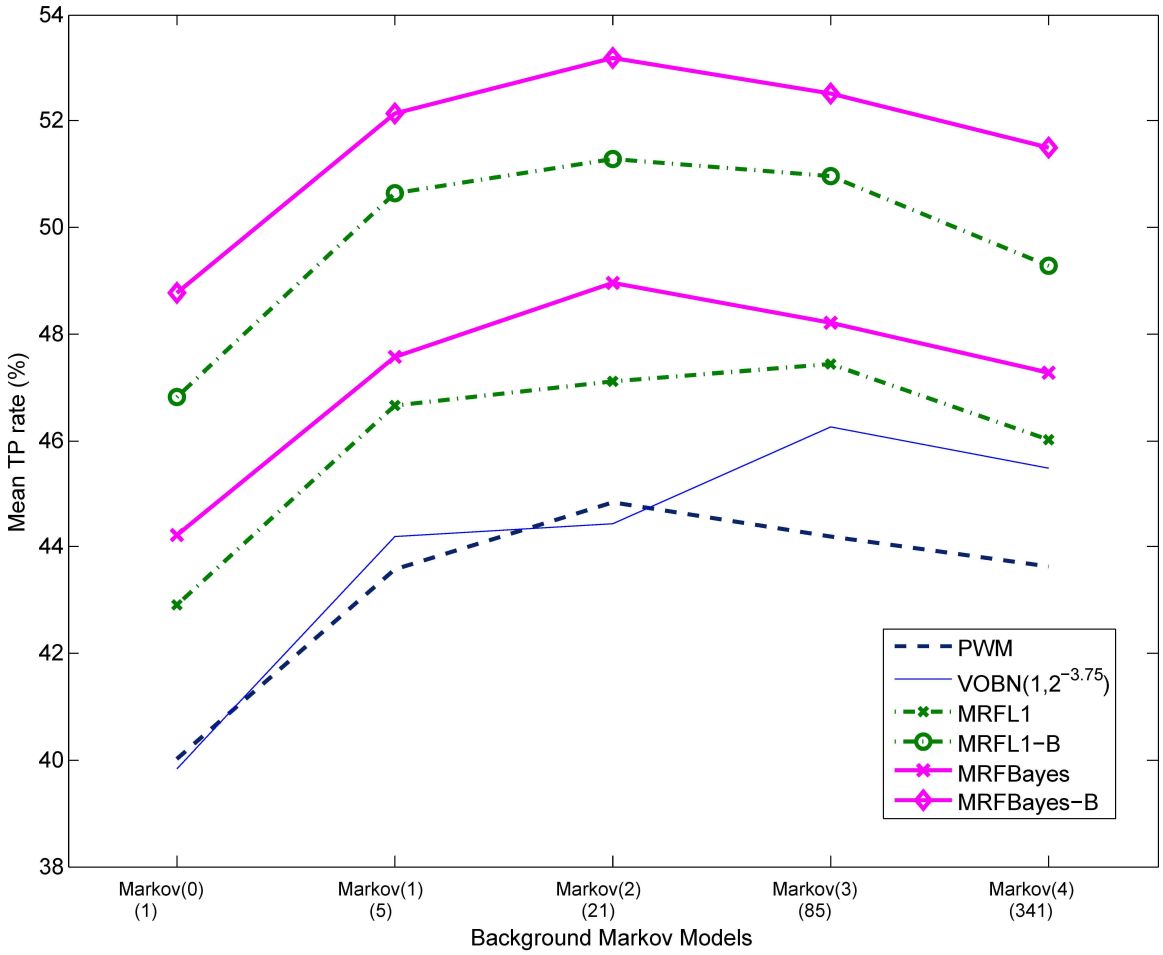

Figure 2.5: Mean TP rates for different fixed order Markov background models at a TN rate of 99.9%. Orders of Markov background models from 0 to 4 are shown on the horizontal axis. The number in the parentheses represents the number of nodes for the corresponding background model.

respectively. These results indicate that the improvement of our MRF model over PWM and VOBN is statistically significant.

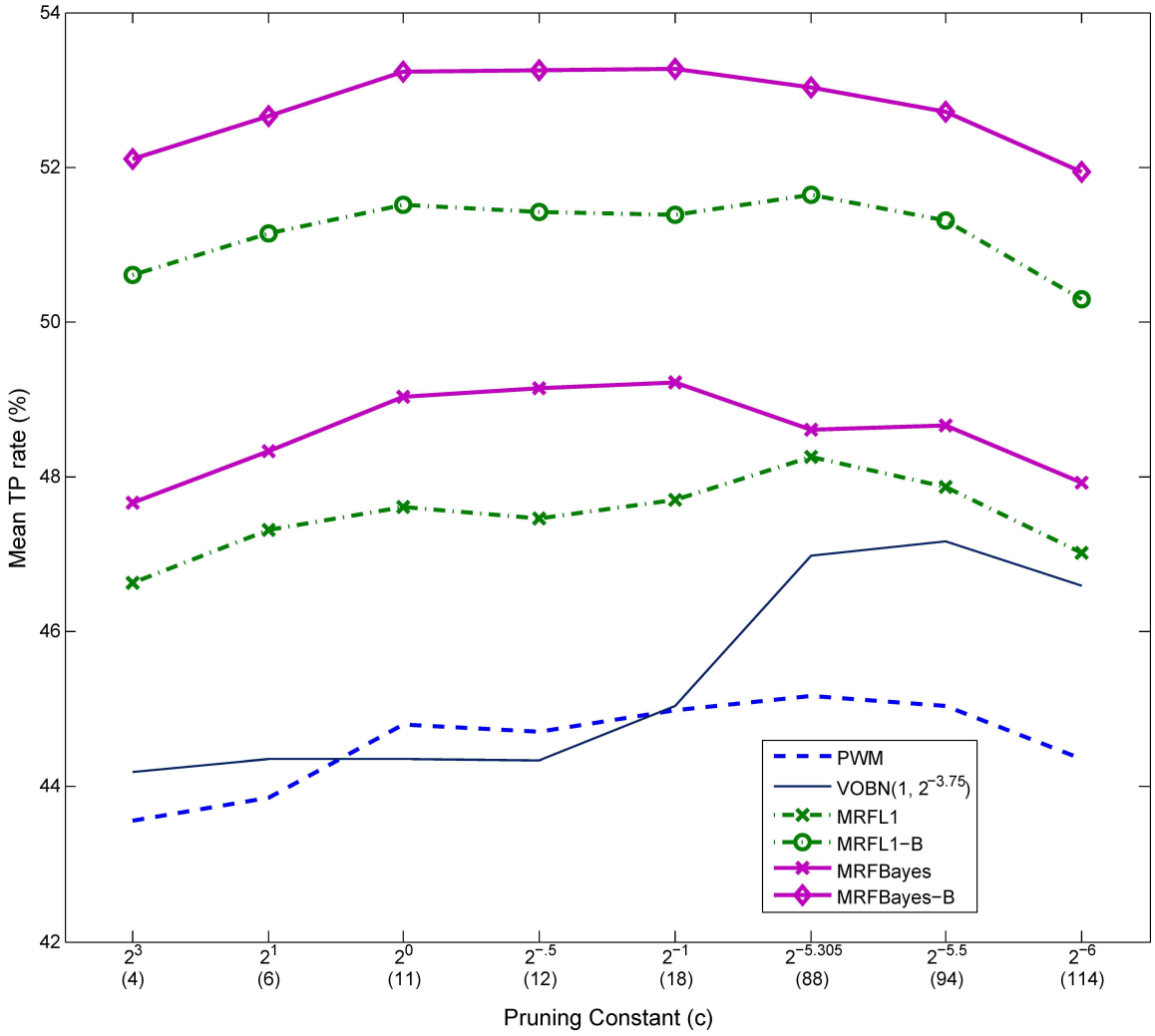

Figure 2.6: Mean TP rates for the VOM background models with an initial order 5 and different pruning constants c at a TN rate of 99.9%. Pruning constants are shown on the x-axis. The average number of nodes in the background model is shown in parentheses for each pruning constant.

2.6.2 Phase 2

Mean TP rates of foreground models with VOM background models constructed from a fifth-order Markov model, with different pruning constants 2^3, 2^1, 2^0, $2^{-0.5}$,

2^{-1}, $2^{-3.305}$, $2^{-5.5}$ and 2^{-6} are shown in Figure 2.6. Mean number of nodes with each pruning constant is shown in brackets, below the pruning constant. The higher the pruning constant value, the lower the number of nodes in the background model, as one can see from Figure 2.6. In general, the lower or higher pruning constants lead to low mean TP rates. This is due to the over-fitting effect of the background model at lower pruning constants and the under-fitting effect at higher pruning constants. According to the results, MRF-based methods outperform both PWM and VOBN with all pruning constants. MRFL1 and PWM both achieve their best TP rate of 48.26% and 45.17%, respectively, with the pruning constant of $2^{-3.305}$, while MRF-Bayes achieves its maximum of 49.22% with the pruning constant of 2^{-1}. Maximum TP rate of 47.17% with the pruning constant of $2^{-5.5}$ of VOBN model agrees with the results in [BGSG$^+$05]. Note that all foreground models achieve comparatively higher accuracy with VOM model as background, because the VOM model is capable of capturing long distance dependencies among base pairs without over-fitting. For comparison purposes we have summarized the maximum mean TP rates in Figure 2.7.

Clearly, the mean TP rate of MRFBayes outperforms all other foreground models PWM, VOBN, and MRFL1 by 4.05%, 2.05%, and 0.96%, respectively. Furthermore, p-values of MRFBayes against PWM and VOBN are 1.33×10^{-14} and 4.26×10^{-5}, respectively. These results provide evidence that the improvement of MRFBayes method is statistically significant.

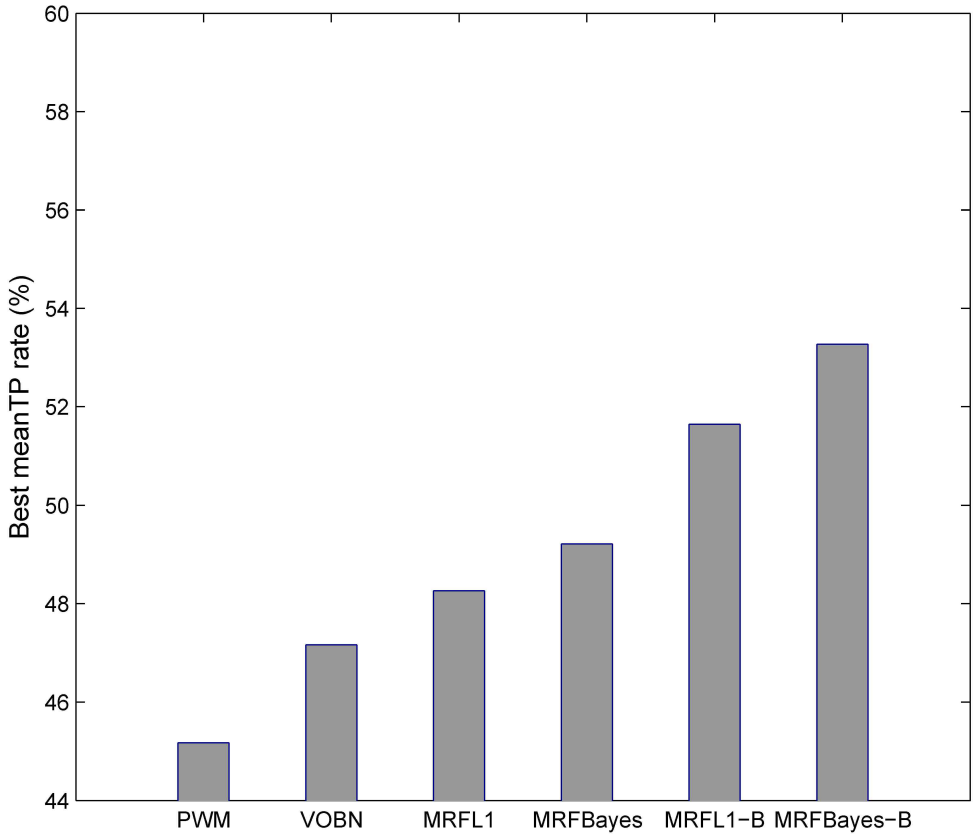

Figure 2.7: Maximum mean TP rates of foreground models with VOM(5, c) as the background model. Refer Table 2.5 to get the corresponding pruning constant of background model (at TN rate 99.9%).

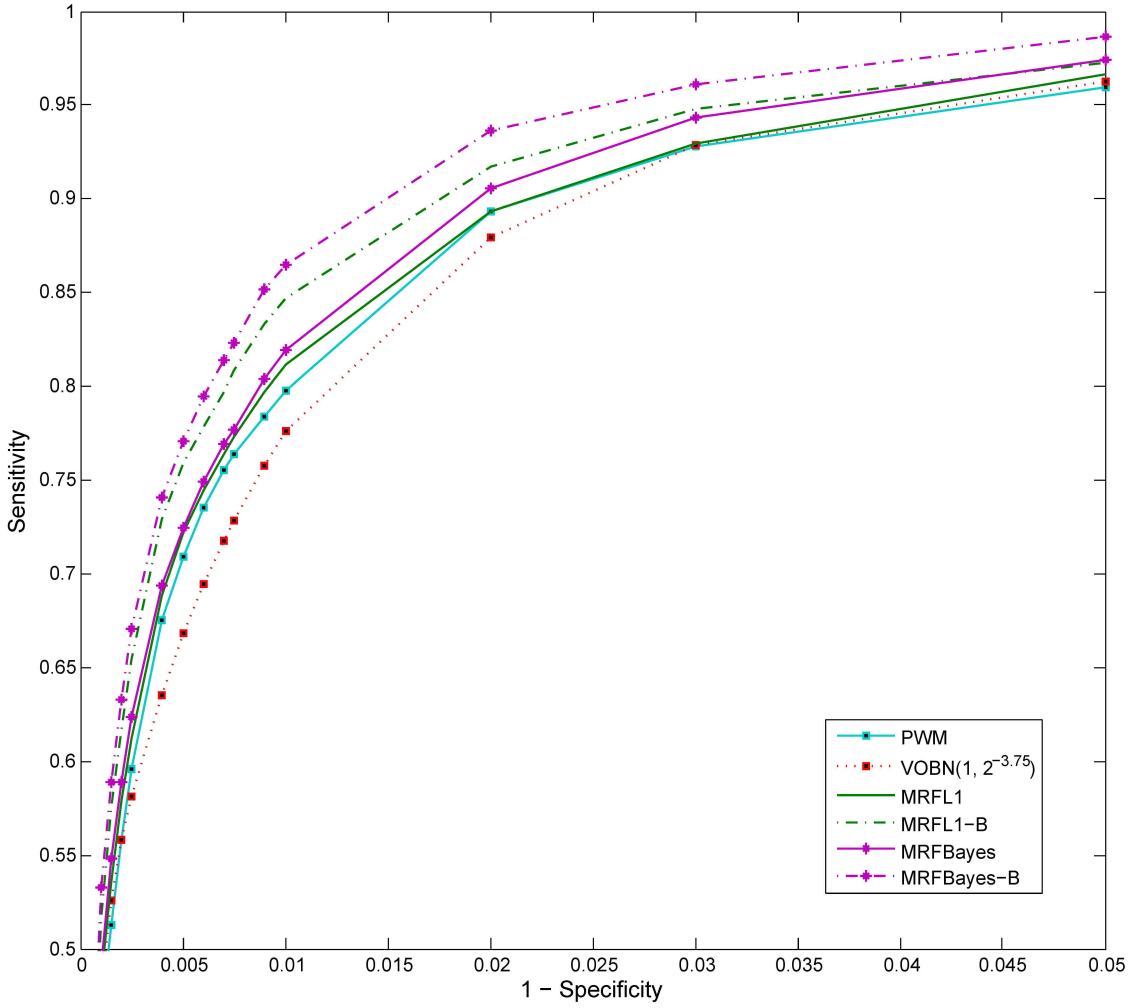

Figure 2.8: The ROC curves of different foreground models. Sensitivity and specificity are the normalized values of mean TP rate and TN rate, respectively.

2.6.3 Phase 3

All previous experiments were conducted with a fixed TN rate of 99.9%. To show how the TP rate of each model is affected by the TN rate, we conducted another set of experiments to draw ROC curves. Figure 2.8 shows the best mean sensitivity (normalized mean TP rate) at different specificity levels (normalized TN rate). As a standard, we used *1-specificity* in x-axis. To select the best mean sensitivity for a given TN rate, we evaluated the mean TP rate with all background models used in the previous two experiments and then selected the maximum mean TP rate. We have listed background models that give the best mean TP rate for each foreground model at different TN rates in Table 2.5.

To obtain ROC curves as shown in Figure 2.8, we changed TN rates from 95% to 99.99%. It is seen that the ROC curves of our MRFL1 and MRFBayes are above the ROC curves of PWM and VOBN(1,$2^{-3.75}$) at all TN rates indicating the significance of our work. To highlight the improvement gained by MRFBayes, we plotted the difference of mean TP rates between MRFBayes and other foreground models, PWM, VOBN, as well as MRFL1, in Figure 2.9. These results indicate that the differences are always positive, implying the improvement of our method is consistent across all TN rates. Compared with VOBN, MRFBayes achieves a maximum improvement of 5.87% in its mean TP rate at the TN rate 99.6%; compared with PWM, MRFBayes achieves a local maximum of 2.2% around 99% and then after 99.25% it is continuously increasing up to a maximum of 7.2%. It is interesting to see that PWM model outperforms VOBN model up to TN rate of 99.75%.

Table 2.5: Background model that gives the best mean TP rate for each foreground model in Figure 2.8

TN (%)	MRFBayes	MRFL1	VOBN(1,$2^{-3.75}$)	PWM
95	Markov(1)	Markov(0)	VOM(5, $2^{3.000}$)	Markov(0)
97	Markov(0)	Markov(0)	VOM(5, $2^{3.000}$)	Markov(0)
98	Markov(0)	Markov(0)	VOM(5, $2^{3.000}$)	Markov(0)
99	Markov(0)	Markov(0)	VOM(5, $2^{3.000}$)	VOM(5, $2^{3.000}$)
99.1	Markov(0)	Markov(0)	Markov(1)	VOM(5, $2^{3.000}$)
99.25	VOM(5, $2^{1.000}$)	Markov(1)	Markov(1)	VOM(5, $2^{-1.000}$)
99.3	VOM(5, $2^{1.000}$)	VOM(5, $2^{1.000}$)	Markov(1)	Markov(2)
99.4	VOM(5, $2^{3.000}$)	Markov(1)	Markov(1)	VOM(5, $2^{1.000}$)
99.5	Markov(1)	VOM(5, $2^{3.000}$)	VOM(5, $2^{1.000}$)	Markov(2)
99.6	Markov(1)	Markov(1)	VOM(5, $2^{1.000}$)	Markov(2)
99.75	VOM(5, $2^{-1.000}$)	VOM(5, $2^{3.000}$)	VOM(5, $2^{-5.305}$)	VOM(5, $2^{-5.305}$)
99.8	VOM(5, $2^{0.000}$)	VOM(5, $2^{-1.000}$)	VOM(5, $2^{-5.500}$)	VOM(5, $2^{-5.305}$)
99.85	VOM(5, $2^{-1.000}$)	VOM(5, $2^{-1.000}$)	VOM(5, $2^{-5.305}$)	VOM(5, $2^{-5.305}$)
99.9	VOM(5, $2^{-1.000}$)	VOM(5, $2^{-5.305}$)	VOM(5, $2^{-5.500}$)	VOM(5, $2^{-5.305}$)
99.98	VOM(5, $2^{0.000}$)	VOM(5, $2^{1.000}$)	VOM(5, $2^{1.000}$)	VOM(5, $2^{1.000}$)
99.99	VOM(5, $2^{1.000}$)	VOM(5, $2^{1.000}$)	VOM(5, $2^{1.000}$)	VOM(5, $2^{1.000}$)

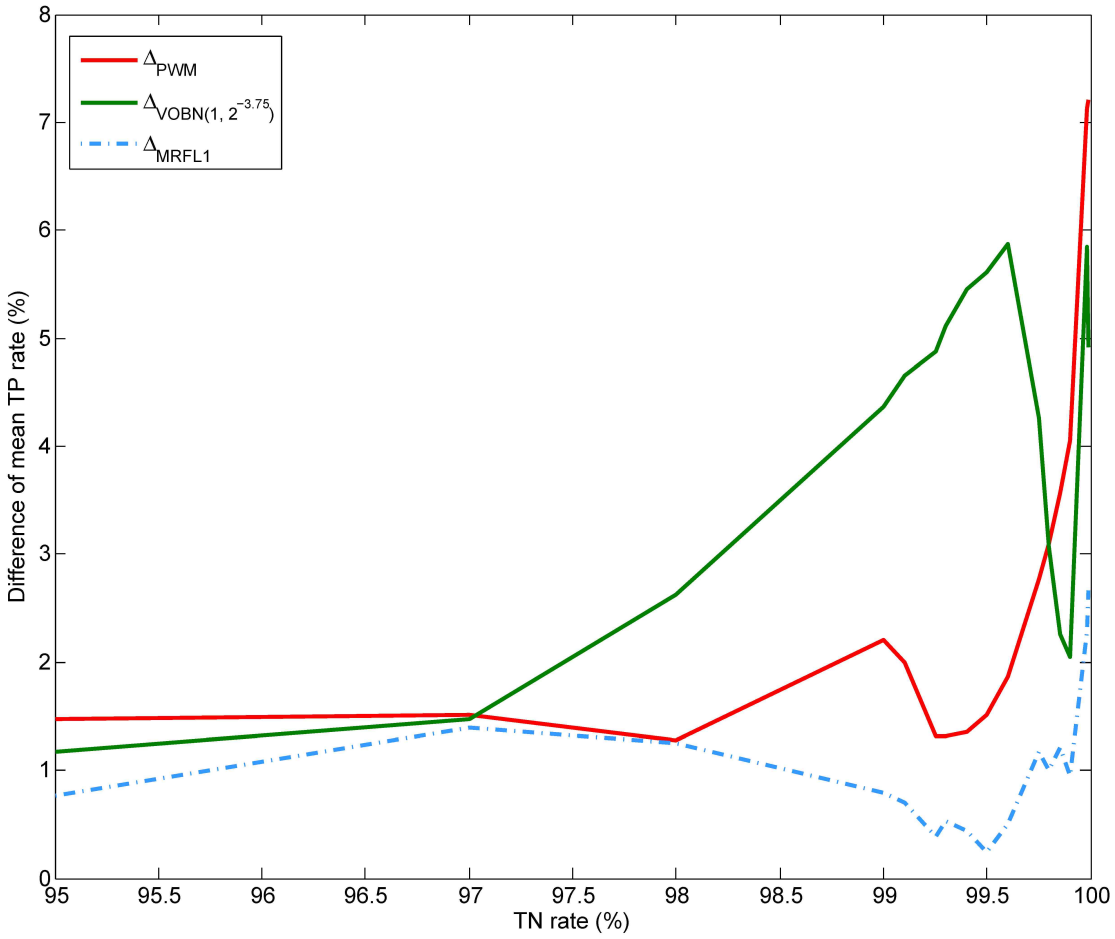

Figure 2.9: Improvement of MRFBayes in mean TP rates relative to PWM, VOBN(1,$2^{-3.75}$), MRFL1. $\Delta_X = MeanTP_{MRFBayes} - MeanTP_X$, where X represents foreground models PWM, VOBN(1,$2^{-3.75}$) and MRFL1.

CHAPTER 3

Pairwise MRF Model and Fast Learning Algorithm

The MRF model we proposed in the previous chapter does not impose any restriction on the structure of the model. Thus, the maximal clique size could grow to the length of the motif. As stated in the previous chapter, the structure of the model is learned by our algorithm using a feature induction method and a ML parameter estimation method with L_1 or Bayesian regularization. Note that the major computational complexity of our algorithm is due to the calculation of partition function Z.

The exact method of evaluating Z has a computational complexity of $4^N \times K$, where N is the length of the TFBS and K is the number of parameters. In the previous chapter, we proposed two levels of feature reduction technique that effectively reduces the value of K in each iteration. These methods perform better when the motif length is comparatively small. However, due to the exponential nature of the computational complexity, the performance is greatly degraded when the motif length gets longer. For instance, a TFBS of length 12 (used in the previous chapter) takes approximately 4-6 hours to learn the structure of the model on a personal computer (PC) having a

CPU speed of 3.4 GHz with a memory size 2GB. It may take much longer time to learn a model with longer motifs.

To address this issue, we propose modeling TFBS using pairwise MRF. In a pairwise MRF, the maximal clique size is two. However, we allow any two nodes to form a clique, which is expected to capture dependencies between any two nodes. The motivation here is that it allows us to develop an approximation method to infer the partition function, which leads to a significantly lower computational complexity. Toward this direction, we adopt an optimization method proposed by Wainwright et al. [WJ06].

The rest of the chapter is organized as follows: Section 3.1 provides a description of restructuring our MRF model to develop a fast learning algorithm. In section 3.2, we describe the formulation of the approximation method, while section 3.3 provides a description of conversion process needed to switch between domains {-1, +1} and {0, 1}. In Section 3.4, we summarized the complete algorithm with approximation method followed by a discussion in section 3.5.

3.1 Pairwise MRF Model

Representing a DNA sequence as a binary valued sequence provides an easy way of incorporating approximation method. Toward this end, we define our configuration set, $\mathcal{A}$, as

$$a_i = 2^{i-1}, i = 1, 2, 3, 4, \tag{3.1}$$

and then represent each a_i as a 4 bit binary number corresponding to the value given by (3.1). This will convert a DNA sequence $x \in \mathcal{X}$, having a length of N, to a $4N$ bit stream. Thus each position of the sequence takes values from the set $\{0,\ 1\}$. The motivation behind this representation is that this will provide an easy way to represent our previously defined atomic features $\mathcal{F}_{\text{atom}}$ as

$$f_{4(i-1)+j} = x_{4(i-1)+j}, i = 1, \cdots, N, j = 1, 2, 3, 4. \tag{3.2}$$

Note that all atomic features still get binary values, because each x_i is either 0 or 1, as we defined earlier. Let the number of atomic features be denoted by n, where $n = 4N$. As explained in the previous chapter, any second-order feature is constructed as a multiplication of two atomic features. Let us denote each second-order feature as

$$f_{i,j} = x_i x_j, i = 1, \cdots, N, i = 1, \cdots, N, \text{ and } i \neq j, \tag{3.3}$$

where i, and j are corresponding atomic feature indexes. Recall the definition we provided for MRF, $G = (V, E)$, where $V = \{1, \cdots, N\}$ is the vertex set and E is a set of undirected edges joining pairs of vertices. Since we include only the first- and second-order features, the number of first- and second-order features is equal to $|V|$ and $|E|$, respectively. Therefore, on a complete graph, the dimension of the weight vector Θ is $n + \binom{n}{2}$. With these notations, we can denote the distribution of sequence x as

$$p(x) = \frac{1}{Z} \exp\Big(\sum_{i=1}^{n} \theta_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \theta_{i,j} x_i x_j\Big). \tag{3.4}$$

For this joint distribution, we obtain the partition function as

$$Z = \sum_{\mathcal{X}} \exp\Big(\sum_{i=1}^{n} \theta_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \theta_{i,j} x_i x_j\Big). \tag{3.5}$$

The distribution given in (3.4) is known as pairwise MRF. The actual complexity of calculation of Z in this model is $2^n \left(\frac{n^2+n}{2}\right)$. Now, we define the log partition function as

$$W(\Theta) = \log Z(\Theta). \tag{3.6}$$

We have deliberately put Θ in W to indicate that the W is a function of Θ to avoid confusion in the next section. With this log partition function, we can rewrite our distribution as

$$p(x) = \exp\Big(\sum_{i=1}^{n} \theta_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \theta_{i,j} x_i x_j - W(\Theta)\Big). \tag{3.7}$$

The reason for introducing this log partition function will be clearer in the next section.

3.2 Approximation Method

In this section, we develop a method to find an approximation for the log partition function for a fully connected pairwise MRF. Wainwright et al. [WJ06] considered the computation of marginal probability of a discrete-valued MRF. Since the exact method of evaluation of partition function is computationally intractable when the number of nodes is large, they proposed a relaxation by using a Gaussian bound on the log partition function and a semidefinite outer bound on the polytope of marginal probabilities. We begin by limiting to the spin variables, where the sequence can

take values from the set {+1, -1} as described by Wainwright et al. [WJ06]. Then we extend their work to solve our original problem in which the sequences take values from the set {0, 1}. With these spin variables, the distribution for a sequence v, can be written as

$$p(v) = \exp\left(\sum_{i=1}^{n} \gamma_i v_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_{i,j} v_i v_j - A(\gamma)\right), \tag{3.8}$$

In this definition of distribution, we consider the log partition function and one can compare this with our similar definition in (3.7). The partition function for this joint distribution is given by

$$A(\gamma) = \log \sum_{\mathcal{V}} \exp\left(\sum_{i=1}^{n} \gamma_i v_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \gamma_{i,j} v_i v_j\right). \tag{3.9}$$

This log partition function maps parameter vectors $\gamma \in \mathbb{R}^d$ to real numbers and is a convex function of γ. This facilitates writing $A(\gamma)$ in a variational fashion as

$$A(\gamma) = \sup_{\eta \in \mathbb{R}} \{\langle \gamma, \eta \rangle - A^*(\eta)\}, \tag{3.10}$$

where $\langle \gamma, \eta \rangle$ is the inner product of vectors γ and η and A^* is an auxiliary function known as the conjugate dual. The dual vector η, where $\eta \in \mathbb{R}^d$, represents the slope of the hyperplane describing a half-space, whereas the dual value $A^*(\eta)$ represents the negative intercept of the hyperplane. Furthermore, with convexity of $A(\gamma)$, we define the conjugate dual formation of $A(\gamma)$ as

$$A^*(\eta) = \sup_{\gamma} \{\langle \eta, \gamma \rangle - A(\gamma)\}, \tag{3.11}$$

where $A^* : \mathbb{R}^d \to \mathbb{R} \cup \{+\infty\}$ [WJ06]. Now, let us define the matrix $M_1[\eta]$ as

$$M_1[\eta] = \begin{bmatrix} 1 & \eta_1 & \eta_2 & \cdots & \eta_{n-1} & \eta_n \\ \eta_1 & 1 & \eta_{1,2} & \cdots & \eta_{n-1} & \eta_{1,n} \\ \eta_2 & \eta_1 & 1 & \cdots & \eta_{n-1} & \eta_n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \eta_{n-1} & \cdots & \cdots & \cdots & \cdots & \eta_{n-1,n} \\ \eta_n & \eta_{n,1} & \eta_{n,2} & \cdots & \eta_{n,n-1} & 1 \end{bmatrix}. \tag{3.12}$$

As explained in [WJ06], the motivation for this representation is that it can be interpreted as the matrix of second-order moments for the vector $(1, v)$, as computed under $p(v)$. Using an entropy bound, Wainwright and Jordan derived a lower bound for the negative dual function as

$$-A^*(\eta) \leq \frac{1}{2} \log \det \left[M_1[\eta] + \frac{1}{3} blkdiag[0, I_n] \right] + n \log \left(\frac{n}{2} \right) \log \left(\frac{\pi e}{2} \right). \tag{3.13}$$

Using the result in (3.13), we can find an upper bound on $A(\gamma)$ as follows.

$$A(\gamma) \leq \max_{\eta \in OUT(K_N), M_i[\eta] \succeq 0} \left[\langle \gamma, \eta \rangle + \frac{1}{2} \log \det \left[M_1[\eta] + \frac{1}{3} blkdiag[0, I_n] \right] \right] + n \log \left(\frac{n}{2} \right) \log \left(\frac{\pi e}{2} \right), \tag{3.14}$$

where $OUT(K_n)$ is the compact convex outer bound on the marginal polytope, $MARG(K_n)$, as described in [WJ06]. We use the notation $M_i[\eta] \succeq 0$ to indicate that matrix $M_i[\eta]$ is positive semidefinite. This problem is strictly concave and has a unique global optimum. The simplest form of (3.14) can be obtained by only imposing the constraint $M_1[\eta] \succeq 0$. Thus, we can use interior point methods specializing

log-determinant problems [VBW98] to solve this. However, the complexity of the generic interior point method is $O(n^6)$. Therefore, the computational complexity is still infeasible for large n. However, we can use a relaxation to the constraint and a dual reformulation so that the final complexity can be reduced to $O(n^3)$, as we describe below.

Note that the term $\left[M_1[\eta] + \frac{1}{3}blkdiag[0, I_n]\right]$ enforces the constraint, $M_1[\eta] \succeq -\frac{1}{3}blkdiag[0, I_n]$, which is a weaker constraint than $M_1[\eta] \succeq 0$. This observation leads to the relaxed problem

$$\max_{\eta} \left[\langle \gamma, \eta \rangle + \frac{1}{2} \log\det \left[M_1[\eta] + \frac{1}{3}blkdiag[0, I_n]\right]\right] + n \log\left(\frac{n}{2}\right) \log\left(\frac{\pi e}{2}\right). \quad (3.15)$$

Next, we introduce a matrix variable $Y = M_1[\eta] + \frac{1}{3}blkdiag[0, I_n]$, and define another matrix, B, with negative weights of γ as

$$B = \begin{bmatrix} 0 & -\gamma_1 & -\gamma_2 & \cdots & -\gamma_{n-1} & -\gamma_n \\ -\gamma_1 & 0 & -\gamma_{1,2} & \cdots & -\gamma_{n-1} & -\gamma_{1,n} \\ -\gamma_2 & -\gamma_1 & 0 & \cdots & -\gamma_{n-1} & -\gamma_n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ -\gamma_{n-1} & \cdots & \cdots & \cdots & \cdots & -\gamma_{n-1,n} \\ -\gamma_n & -\gamma_{n,1} & -\gamma_{n,2} & \cdots & -\gamma_{n,n-1} & 0 \end{bmatrix}. \quad (3.16)$$

By defining B and Y in this way, we can simply write the $\langle \gamma, \eta \rangle$ as $\frac{1}{2}\langle -B, Y \rangle$, where $\frac{1}{2}\langle -B, Y \rangle$ is the *Frobenius inner product* or the component-wise inner product of matrix $-B$ and Y, evaluated as $trace(-B^T Y)$ or in our case $trace(-BY)$ since the matrices B and Y are symmetric. The Frobenius product is also evaluated as

the sum of the entries of the *Hadamard product* given by $\sum_i \sum_j B_{i,j} Y_{i,j}$. With these notations, the relaxed log determinant problem can be written as

$$\max_{Y \succeq 0} \frac{1}{2} \left[\langle -B, Y \rangle + \log\det Y \right] + n \log\left(\frac{n}{2}\right) \log\left(\frac{\pi e}{2}\right), \tag{3.17}$$

where the $diag(Y) = d = [1, 4/3, \cdots, 4/3]^T$. Dropping the constant terms and multiplication factor of $\frac{1}{2}$, we can define a simplified version of (3.17) as

$$\max_{Y \succeq 0} \left[\langle -B, Y \rangle + \log\det Y \right]. \tag{3.18}$$

This optimization problem can be converted to an unconstrained optimization problem by obtaining the Lagrangian dual as follows:

$$Q(\lambda) = -(n+1) - \log\det\left[B + diag(\lambda)\right] + \langle diag(\lambda), diag(d) \rangle, \tag{3.19}$$

where, $\lambda \in \mathbb{R}^{n+1}$ are the Lagrange multipliers associated with the linear constraints $diag(Y) = d$. And also the gradient of Q, $\nabla Q(\lambda) = -diag[B + diag(\lambda)]^{-1} + d$ and Hessian $\nabla^2 Q(\lambda) = -\left([A + diag(\lambda)]^{-}1\right) \odot \left([A + diag(\lambda)]^{-}1\right)$, where $\odot$ denotes Hadamard product, are obtained by using properties of matrix derivatives. Given the optimum value of λ^*, we can find the solution to Y as $Y^* = [B + diag(\lambda)]^{-1}$. The main assumption of this problem is that the matrix $[B + diag(\lambda)] \succeq 0$. Otherwise, we cannot solve the above problem since the (3.19) involves a log term of the determinant of $[B + diag(\lambda)]$.

It is not difficult to show that, by adding a proper constant values to the diagonal of the matrix B in (3.16), we can always satisfy this condition for any given matrix B.

Proof: Let us denote the minimum eigenvalue of any weight matrix B, as λ_{min}. Since the matrix B is real and symmetric, using eigenvalue decomposition we can always write B as

$$B = U\Lambda U^T$$

where U is an orthogonal matrix and Λ is real and diagonal matrix having the eigenvalues of B on the diagonal. Furthermore, we define a constant σ such that, $\sigma \geq -\lambda_{min}$. Then consider the matrix $B^* = B + \sigma I$,

$$\begin{aligned} B^* &= U\Lambda U^T + \sigma I \\ &= U\Lambda U^T + \sigma U U^T \\ &= U(\Lambda + \sigma I)U^T \end{aligned}$$

Let $\Lambda + \sigma I = \tilde{\Lambda}$. With this notation we can write B^* as

$$B^* = U\tilde{\Lambda}U^T.$$

By comparing this equation with eigenvalue decomposition, we notice that the diagonal elements of matrix $\tilde{\Lambda}$ represent the eigenvalues of B^*. Furthermore, according to our selection of σ, all the values of this diagonal matrix $\tilde{\Lambda}$ are non-negative because $\sigma + \lambda_{min} \geq 0$. This concludes our proof that, B^* is positive semidefinite. ■

With this representation of B^*, let us express the term $\langle -B^*, Y \rangle$ as

$$\begin{aligned} \langle -B^*, Y \rangle &= \langle -(B + \sigma I), Y \rangle \\ &= \langle -B, Y \rangle + \langle -\sigma I, Y \rangle \\ &= 2\langle \gamma, \eta \rangle + \langle -\sigma I, diag(d) \rangle. \end{aligned} \tag{3.20}$$

By substituting (3.20) into (3.15), and introducing a matrix variable $Y = M_1[\eta] + \frac{1}{3}blkdiag[0, I_n]$, we get the relaxed optimization problem with B^* as follows:

$$\max_{\eta} \frac{1}{2}\left[(\langle -B^*, Y\rangle - \langle -\sigma I, diag(d)\rangle) + \log\det Y\right] + n\log\frac{n}{2}\log(\frac{\pi e}{2}) \tag{3.21}$$
$$= \max_{\eta} \frac{1}{2}\left[\langle -B^*, Y\rangle + \log\det Y\right] - \frac{1}{2}\langle -\sigma I, diag(d)\rangle + n\log\left(\frac{n}{2}\right)\log\left(\frac{\pi e}{2}\right).$$

We drop all the constants and multiplication factor of $\frac{1}{2}$ to obtain the simplified version of (3.21) as

$$\max_{\eta} \left\{\langle -B^*, Y\rangle + \log\det Y\right\}. \tag{3.22}$$

Note that (3.18) and (3.22) have the same format, but (3.22) has B^* instead of B. Therefore, we can use the Lagrangian dual function we derived in (3.19) to solve the optimization problem by replacing B with B^*. Recall that B^* is positive semidefinite. This implies that the term $\det[B^* + diag(\lambda)]$ is also positive semidefinite (λ is always ≥ 0), and this alleviates the previous problem we had in (3.19). This unconstrained problem can be solved with the complexity of $O(n^3)$, using Newton's method or any other gradient method.

The main complexity is due to the inversion of $(n+1) \times (n+1)$ matrix and finding the minimum eigenvalue of the matrix B^*. There are some approximation methods to calculate the minimum eigenvalue of a matrix. In our case we do not need the exact value of the minimum eigenvalue but a lower bound, because we need to satisfy the condition $\sigma \geq -\lambda_{min}$. So far, we have completely formulated the approximation method to work with spin variables with any given weight matrix. We now describe the way of using these results to solve our original problem with {0, 1} valued variables.

3.3 Model Conversion

In this section, we make the relation between spin variables and $\{0, 1\}$ valued variables. First, let us define the relation between variables v_i and x_i as

$$v_i = 2x_i - 1 \tag{3.23}$$
$$x_i = \frac{1}{2}(v_i + 1).$$

This definition provides an easy way of going back and forth between models, by converting $\{0, 1\}$ representation of variable x_i into $\{-1, +1\}$ representation of variable v_i and vice versa. We begin with our original distribution defined in (3.7). Then we substitute the expression for x_i using (3.23) to the exponential expression in (3.7),

$$\begin{aligned}
\sum_{i=1}^{n}\theta_i x_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}x_i x_j &= \sum_{i=1}^{n}\theta_i\frac{1}{2}(v_i+1) + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}\frac{1}{4}(v_i+1)(v_j+1) \qquad (3.24)\\
&= \sum_{i=1}^{n}\frac{1}{2}\theta_i v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}(v_i+v_j) + \\
&\qquad \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_i v_j + \sum_{i=1}^{n}\frac{1}{2}\theta_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}\\
&= \sum_{i=1}^{n}\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n}\theta_{i,j}\Big)v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_i v_j + \\
&\qquad \sum_{i=1}^{n}\frac{1}{2}\theta_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}\\
&= \sum_{i=1}^{n}\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n}\theta_{i,j}\Big)v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_i v_j + \xi
\end{aligned}$$

where $\xi = \sum_{i=1}^{n}\frac{1}{2}\theta_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}$, and we use the fact that $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}(v_i + v_j) = \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\theta_{i,j}v_i$, that is proved as follows.

Proof:

$$\begin{aligned}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}(v_i+v_j) &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}v_j \\ &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}v_i + \sum_{j=2}^{n}\sum_{i=1}^{j-1}\theta_{i,j}v_j \\ &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}v_i + \sum_{i=2}^{n}\sum_{j=1}^{i-1}\theta_{j,i}v_i\end{aligned}$$

Since we consider an MRF, edges are undirected, Thus, $\theta_{i,j} = \theta_{j,i}$. Using this representation, we can write $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}(v_i+v_j)$ as

$$\begin{aligned}\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}(v_i+v_j) &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\theta_{i,j}v_i + \sum_{i=2}^{n}\sum_{j=1}^{i-1}\theta_{i,j}v_i \\ &= \sum_{i=1}^{n}\sum_{j=1,j\neq i}^{n}\theta_{i,j}v_i.\end{aligned}$$

■

Substituting (3.24) into (3.7), we obtain

$$\begin{aligned}p(v;\Theta) = \exp\Bigg[&\xi + \sum_{i=1}^{n}\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n}\theta_{i,j}\Big)v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_iv_j \\ &- \log\left(\sum_{\mathcal{V}}\exp\left(\xi + \sum_{i=1}^{n}\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n}\theta_{i,j}\Big)v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_iv_j\right)\right)\Bigg] \\ = \exp\Bigg[&\sum_{i=1}^{n}\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n}\theta_{i,j}\Big)v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_iv_j \\ &- \log\left(\sum_{\mathcal{V}}\exp\left(\sum_{i=1}^{n}\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n}\theta_{i,j}\Big)v_i + \sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\frac{1}{4}\theta_{i,j}v_iv_j\right)\right)\Bigg]\end{aligned} \tag{3.25}$$

Since we have used the variable conversion from {0, 1} domain to {-1, +1} domain, distribution (3.25) should represent exactly the same distribution we defined in (3.8). Thus, we can get the relationships as follows:

$$\frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n} \theta_{i,j}\Big) = \gamma_i$$

$$\frac{1}{4}\theta_{i,j} = \gamma_{i,j}$$

By substituting these into (3.7), we can convert the distribution to

$$p(v;\gamma) = \exp\left[\gamma_i v_i + \gamma_{i,j} v_i v_j - \log\left(\sum_{\mathcal{V}} \exp\left(\gamma_i v_i + \gamma_{i,j} v_i v_j\right)\right)\right]. \tag{3.26}$$

From our derivation, $A(\gamma)$ and $W(\Theta)$ are related by

$$W(\Theta) = A(\gamma) + \xi. \tag{3.27}$$

Using the log partition function $W(\Theta)$, we can easily evaluate the original partition function Z as $\exp(W(\Theta))$. Thus, we can use the following set of relationships to go back and forth between two distributions.

$$\gamma_i = \frac{1}{4}\Big(2\theta_i + \sum_{j=1,j\neq i}^{n} \theta_{i,j}\Big) \tag{3.28}$$

$$\gamma_{i,j} = \frac{1}{4}\theta_{i,j}$$

$$\theta_i = 2\Big(\gamma_i - \sum_{j=1,j\neq i}^{n} \theta_{i,j}\Big)$$

$$\theta_{i,j} = 4\gamma_{i,j}$$

3.4 Algorithm

In the previous sections, we explained the approximation procedure to estimate a value for $A(\gamma)$ by considering spin variables. Then we presented the conversion pro-

cedure that facilitates estimating a value for our original partition function, Z, using the results we derived. Recall that in Chapter 2, in each iteration of the numerical optimization procedure, we needed to evaluate the value of Z. Therefore, in each iteration of the main optimization loop, we need to run another sub iteration procedure to find the approximation value for Z. As shown in our discussion, the computational complexity of approximate evaluation of Z is in the order of $O(n^3)$, with the exact complexity being the $2^n \left(\frac{n^2+n}{2}\right)$. Thus, this approximation procedure drastically reduces the overall complexity of our algorithm.

Since our original model works with 0, 1 valued sequences, we use (3.28) to go back and forth between distributions in {0,1} domain and {-1, +1} domain. As explained earlier, we allow all the first- and second-order features to form the model. This eliminates the need for the feature selection algorithm we described in Section 2.3.4. We can use any of the two types of regularization methods, as discussed in the previous chapter in the main optimization loop. The complete algorithm is summarized in Table 3.1.

3.5 Results and Discussion

As described earlier, the main computational burden in learning and inferring the model is in calculating the partition function, Z. As described in Chapter 2, the exact method of evaluating Z has a computational complexity of $4^N \times K$, where N is the length of the TFBS and K is the number of parameters. However, due to the exponential nature of the computational complexity, the performance is greatly

Table 3.1: Algorithm with L_1 regularization

Begin
  Initialization
    Add all first and second order features
    Initialize weight parameters
  EndInitialization

  Optimization
    repeat
      SubOptimization
        Compute matrix B^* using (3.28)
        Do Optimization to compute λ^* (3.19)
        Compute Y^* as $Y^* = [B^* + diag(\lambda^*)]^{-1}$
        Computer $A(\gamma)$ using (3.17) and (3.18)
        Compute Z using (3.28)
      EndSubOptimization
      Do work in main optimization
    until Main optimization complete
  EndOptimization
  Save the Model
End

degraded when the motif length gets longer. For an example, a TFBS of length 12 took approximately 4-6 hours to learn the structure of the model on a personal computer (PC) having a CPU speed of 3.4 GHz with a memory size 2GB. It may take much longer time to learn a model with longer motifs.

However, the recent interest of research requires dealing with motifs having longer lengths [Bai11, HBS+10, GPGK13, HZS+11, YMF+14, PS14, FBS08, RB07] that requires more efficient algorithms to reduce the computation complexity of structure learning. Capitalizing on this idea, we developed an approximation method to evaluate the partition function in this chapter. The computational complexity of approximate evaluation of Z is on the order of $O(n^3)$, with the exact complexity being

the $2^n \left(\frac{n^2+n}{2}\right)$. Thus, this approximation procedure drastically reduces the overall complexity of our algorithm.

To demonstrate the performance of the novel approximation method presented in this chapter over the exact method developed in Chapter 2, we tabularized the complexity of both algorithms over various motif lengths. Table 3.2 summarizes the complexity of both algorithms for a fixed K of 100. As one can clearly see, the exact method becomes quickly unusable when motif lengths get longer and longer. Recall the observation we had from Chapter 2 where it took 6 hours to learn the model for motifs with length 12. If we were to predict the learning time for a model with motif length of 20 based on this observation, it would take 44.89 years with the exact method. In comparison, it would take 27.77 hours with the approximation method.

To highlight the performance improvement further, we plotted the computational complexity of approximation method over exact method in Figure 3.1 and improvement of the performance in Figure 3.2. The improvement of the performance was calculated as $\frac{t_{exact}}{t_{approx}}$. We also included logarithmic scale for the y-axis in bottom figures to easily visualize the differences. As one can clearly see, the improvement over the performance is very impressive with longer motifs. This is highly desirable when dealing with problems that demand higher motif lengths, such as *de novo* TFBSs identification, as we further describe in the next section.

Table 3.2: Computational complexity comparison of MRF-approximation method over MRF-exact method.

N	MRF-Approx	MRF-Exact
1	1	400
2	8	1600
3	27	6400
4	64	25600
5	125	102400
6	216	409600
7	343	1638400
8	512	6553600
9	729	26214400
10	1000	104857600
11	1331	419430400
12	1728	1677721600
13	2197	6710886400
14	2744	26843545600
15	3375	1.07374E+11
16	4096	4.29497E+11
17	4913	1.71799E+12
18	5832	6.87195E+12
19	6859	2.74878E+13
20	8000	1.09951E+14

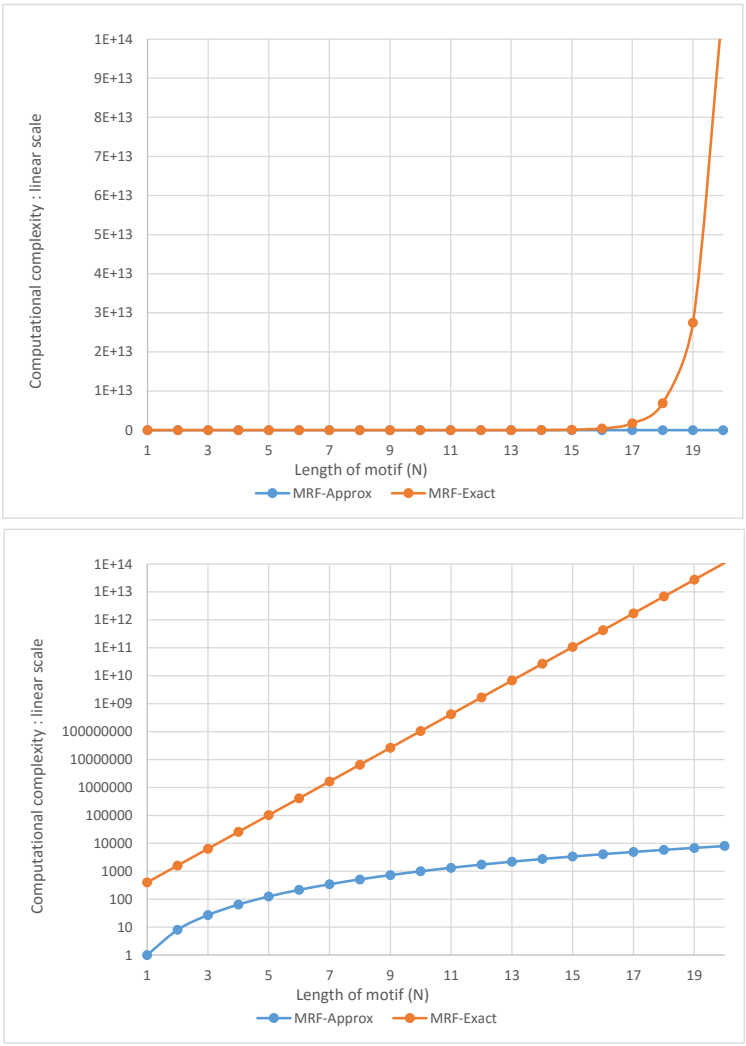

Figure 3.1: Variation of computational complexity of approximation method and exact method with different motif lengths (N). Top: linear scale, Bottom: log scale

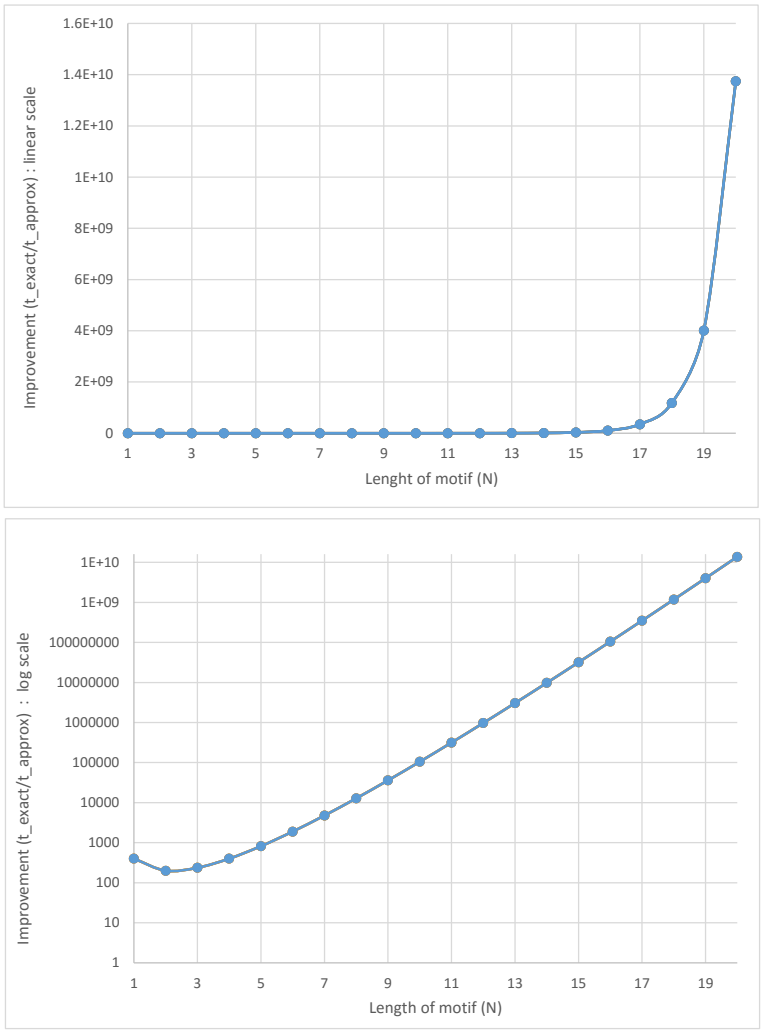

Figure 3.2: Improvement of approximation method over exact method for various motif lengths (N). Improvement was calculated as t_{exact}/t_{approx}. Top: linear scale, Bottom: log scale

3.5.1 *De Novo* Discovery of TFBSs

So far, we have used a set of training data that contains experimentally verified TFBSs to train models. This method falls into the category of *supervised learning*, where we use a training data set to learn the model. Now we are considering the possibility of applying our proposed MRF-based model in solving the problem of identifying TFBS in the *unsupervised learning* domain where no training data set is used. These methods are used for *de novo* discovery of TFBSs and they play a vital role in identifying novel TFBSs that have not been discovered experimentally.

In general, the unsupervised learning process takes a set of sequences as input and discovers the patterns shared by some of the sequences. Usually, unsupervised learning is harder than supervised learning because the space of possible patterns is much larger, and also the patterns to be discovered are not required to be in any of the sequences. Thus, the unsupervised learning algorithms simultaneously look for a cluster of input sequences and patterns common to the members of this cluster. Now, let us briefly describe the work has been done in this direction.

The novel TFBSs identification problem can be formulated as a process where it outputs the locations and sites as putative binding sites, within a set of regulatory regions that are known to be regulated by the same transcription factor(s). The problem itself is complicated because we need to infer both the pattern of the motifs and their locations in the input sequences. Since the problem tries to model the similarity among segments of nucleotides known as oligonucleotides (a short nucleotide sequence), two main approaches have been introduced to represent oligos (short term

for oligonucleotide), align consensus sequences or express them with a profile such as PWM, PSSM, or PFM. As we will see shortly, the latter is the one where we have room to improve the performance of identifying novel TFBS.

The fundamental problem of identifying novel TFBS is that we do not know the location. This naturally inclined us to do an exhaustive search with different locations and became the first approach to the problem [GES85, WAG84]. The main overhead of this method is the exponential search space. This complexity was later significantly reduced by using proper indexing structures such as suffix trees [Gus97]. Later in this row, various algorithms based on exhaustive search with allowed mutations were developed such as SMILE [MS00] and Weeder [PMP04]. All of these algorithms are based on consensus-based representation of motifs. Since the consensus representation is not flexible enough to describe a motif [Sto00], a different approach of alignment based methods was proposed.

Basically, these methods try to align a given set of sequences with some scoring schemes to fit into the best profile. But it has been proven that finding the best profile is an NP-hard problem [AAS00]. This implies that the determination of the best profile is computationally infeasible, no matter what scoring scheme is used. To alleviate this problem, some *heuristic* methods are used to prune the search space. As an example, *Consensus* [SH89], is a method based on greedy heuristics. Methods such as BLAST [AGM$^+$90], FASTA [PL88] all fall in this category.

Another way of finding the best alignment profile is to use a statistical method to infer the location of the motifs and the profile jointly. The Bayesian method and several related Gibbs sampling algorithms for motif discovery fall into this cat-

egory [LNL95]. Instead of using the Bayesian method, Lawrence et al. [LR90] used the maximum likelihood method that relies on EM algorithm to estimate the location of PWM of the motifs. The original algorithm has a restriction of one TFBS per sequence. Bailey et al. relaxed this restriction by introducing a method called MEME [BE94, BE95].

3.5.2 MRF Approach to *De Novo* Identification of TFBSs

As discussed in the previous section, profile-based methods find more interest because they are capable of modeling motifs better than the consensus-based methods. The basic problem of profile representation is that they assume a statistical independence among base pairs. As we described earlier, recent studies show that there are some dependencies [BJC02, BLFS01, WGRP99, MS01, BPQ+06, UMFK02, OPLS05], and hence this representation does not model the motif well. Since we have developed an MRF model capable of capturing more dependencies among base pairs, we believe that if we apply our MRF to *de novo* identification of TFBSs, it will provide better performance than conventional ML or Bayesian methods based on PWM.

As we have shown earlier, the MRF model can be considered as a better representation of modeling motifs. We use this fact to investigate the possibility of incorporating our MRF model with EM algorithm to use in novel TFBS identification. Bailey et al. have done some work in this direction using existing profile base modeling. We hope that we can improve the performance by using MRF to model the motif.

Specifically, we can treat the locations of motif as missing data and develop an EM algorithm that can handle missing data and maximize the likelihood function of the input data sequences based on a pairwise MRF model for TFBSs, and a PWM or fixed order Markov model for background. Since the EM algorithm runs in an iterative fashion, and in each iteration we need to maximize the likelihood function to estimate model parameters, the computational complexity of the algorithm may be large. However, using the fast optimization algorithm we developed here, we expect that the speed of the EM algorithm can be increased dramatically.

CHAPTER 4

Fast Proximal Gradient Optimization of the Empirical Bayesian Lasso for Multiple Quantitative Trait Locus Mapping

Complex quantitative traits are influenced by many factors, including those of many quantitative trait loci (QTLs), epistatic effects involving more than one QTL, environmental effects and effects of gene-environment interactions. We develop here a fast proximal gradient optimization algorithm with empirical Bayesian Lasso (EBlasso) method that uses a high-dimensional sparse regression model to infer the QTL effects from a large set of possible effects. The new algorithm outperforms state-of-the-art algorithms in terms of power of detection (PD) and false discovery rate (FDR), and capable of handling a relatively large number of possible QTLs.

4.1 Background

Such complex quantitative traits are usually controlled by multiple quantitative trait loci (QTLs) along with environmental factors. Quantitative trait QTL mapping goal

identifies multiple genomic loci that are associated with the traits and estimates the genetic effects of these loci, possibly including some of the main effects, gene-gene interactions (epistatic effects) and gene-environment interactions. Although technology advancement in molecular genotyping has made high density genomic markers available, including these markers with their possible interactions in a single QTL model, it leads to a large number of model variables, typically much larger than the sample size. This not only involves huge computation, which is challenging to existing QTL mapping methods, but it also may reduce the power of detection (PD) and maybe increase the false discover rate (FDR). As described in Chapter 1, variable selection and shrinkage operator are two techniques often used in the inference of such high dimensional QTL models.

Phenotypic variation is best explained through variable selection, which identifies a subset of all possible genetic effects [LLF$^+$09]. This usually occurs by stepwise forward selection or backward elimination along with selection criteria such as the Bayesian information criterion (BIC) [Sch78] to restrict the model space. Shrinkage methods, like Lasso [Tib96], elastic net [ZH05] and Bayesian Lasso [YX08, PC08] have all variables in the model, but use a penalty function of the variables or appropriate hierarchical prior distributions for the variables to shrink most variables toward zero. An approach that has gained a great deal of attention is the Markov Chain Monte Carlo (MCMC) simulation-based Bayesian shrinkage approach [OS09] that has been applied to multiple QTL mapping [YX08, WYL$^+$05, Xu03]. MCMC simulation is quite computationally intensive (see Chapter 2 herein); with a large number of effects in the model, it can also be time consuming. So more efficient methods [HWIB08,

YB09] have been developed to reduce computational burden for the Bayesian QTL models.

An efficient empirical Bayesian Lasso (EBlasso) algorithm has been developed, from a two-level hierarchical model that has normal and exponential priors (EBlasso-NE) or a three-level hierarchical model with normal, exponential, and Gamma priors (EBlasso-NEG) [CHX11, HXC13]. Also developed was an empirical Bayesian elastic net (EBEN) with the use of a two-level hierarchical model that had normal and generalized gamma priors [HXC14a] for multiple QTL mapping. EBlasso and EBEN both outperform other methods, including Lasso and MCMC-based Bayesian shrinkage in terms of PD and FDR. Whole-genome QTL mapping [HXC14b] and pathway-based genome-wide association study (GWAS) [HMVC14] have had EBlasso applied, with linear regression models with millions of variables inferred. Unfortunately, the accuracy and computational speed may not be optimal for both methods, as they are optimized by a greedy coordinate ascent algorithm.

Taking advantage of recent advancements in convex optimization, we developed a fast proximal gradient algorithm for the EBlasso-NE method (referred to as EBlasso hereon). Because of the advanced optimization algorithm and other techniques, this new method is quite efficient, in comparison with the previous one that was optimized by coordinate ascent algorithm. Simulations have shown that the proximal gradient algorithm provides better PD and FDR than did the coordinate ascent method, and does so with computational time that is considerably reduced.

4.2 Linear Model of Multiple QTLs

Let y_i be the phenotypic value of a quantitative trait of the i^{th} individual in a mapping population. Assume we observe y_i, $i = 1, \cdots, n$, of n individuals and collect them into a vector $y = [y_1, y_2, \cdots, y_n]^T$. In these n individuals, further suppose p environmental covariates are observed and q genetic markers genotyped. Let covariate l and genotype of marker j of individual i be x_{Eil} and x_{Gij}, respectively. We define $X_{Ei} = [x_{Ei1}, x_{Ei2}, \cdots, x_{Eip}]^T$ and $X_{Gi} = [x_{Gi1}, x_{Gi2}, \cdots, x_{Giq}]^T$. Then we have the following linear regression model for y:

$$y = \mu + X_G\beta_G + X_E\beta_E + e, \tag{4.1}$$

where μ is the population mean, vectors β_G and β_E represent the genetic effects of all markers and environmental effects; matrices $X_G = [X_{G1}, X_{G2} \cdots X_{Gn}]^T$ and $X_E = [X_{E1}, X_{E2} \cdots X_{En}]^T$ are the corresponding design matrices of different effects, and e is the residual error that follows a normal distribution with zero-mean and covariance $\sigma_0^2 I$.

X_G, the design matrix, depends on a specific genetic model, for which we adopt the widely used Cockerham genetic model [Coc54]. It defines the values of a marker effect as -0.5 and 0.5 for two genotypes in a back cross design; -1, 0 and 1 for three genotypes having an additive effect; and -0.5 and 0.5 for homozygotes and heterozygotes having a dominance effect in an intercross (F2) design. For the sake of simplicity we only look at additive effects in (4.1). However, the method developed here is also applicable to a model with dominance effects. Epistatic effects can also be incorpo-

rated into (4.1) as done in [CHX11]. However, for the ease of presentation, we use model (4.1) throughout the chapter.

Defining $\beta = [\beta_0, \beta_G^T, \beta_E^T]^T$, $X = [X_G, X_E]$, we can write (4.1) in a more compact form:

$$y = \mu + X\beta + e, \tag{4.2}$$

Given q markers with additive main effects and p environmental covariates, the size of matrix X is $n \times m$ where $m = p + q$. Only a small portion of marker effects are expected to be QTL effects, implying that β is a sparse vector, and thus most elements of β are zero. We thereby develop an efficient EBlasso algorithm to infer sparse β from (4.2) (see the next two sections).

4.3 Prior and Posterior Distribution

There are three unknown parameters in model (4.2): μ, σ_0^2, and β. We are mainly interested in β, but μ and σ_0^2 need to be estimated so that we can infer β. A noninformative uniform prior is assigned to μ and σ_0^2, i.e., $p(\mu) \propto 1$ and $p(\sigma_0^2) \propto 1$. Then a two-level hierarchical model is assumed for β. Elements of β are denoted as β_j, $j = 1, 2, \cdots, m$. At the first level, β_j, $j = 1, 2, \cdots, m$ follow independent normal distributions with mean zero and unknown variance τ_j : $\beta_j \sim N(0, \tau_j)$. Then at the second level, τ_j, $j = 1, 2, \cdots, m$, follow independent exponential distribution with a common parameter λ: $p(\tau_j) = \lambda \exp(-\lambda \tau_j)$. For a given λ, the distribution of β_j is the Laplace distribution: $p(\beta_j) = \sqrt{\frac{\lambda}{2}} \exp(-2\sqrt{\lambda}|\beta_j|)$; this is known to encourage the shrinkage of β toward zero [Tib96].

Let $\tau = [\tau_1, \tau_2, \cdots, \tau_m]^T$. Our EBlasso method first estimates parameters τ, σ_0^2 and μ and then finds the posterior distribution of β based on the estimated parameters. The posterior distribution of μ, β, τ and σ_0^2 is given by

$$p(\mu, \beta, \tau, \sigma_0^2|y) \propto p(y|\mu, \beta, \sigma_0^2)p(\mu)p(\beta|\tau)p(\tau|\lambda)p(\sigma_0^2). \tag{4.3}$$

The marginal posterior distribution of μ, τ, and σ_0^2 can then be written as

$$p(\mu, \tau, \sigma_0^2|y) = \int p(\mu, \beta, \tau, \sigma_0^2|y)d\beta. \tag{4.4}$$

We collect all parameters that need to be estimated as $\theta = (\mu, \sigma_0, \tau)$. The log marginal posterior distribution of θ can be found from (4.4) as follows:

$$L(\theta) = -\frac{1}{2}\left[\log|C| - \frac{1}{2}(y-\mu)^T C^{-1}(y-\mu)\right] - \lambda\sum_{j=1}^{m}\tau_j + constant \tag{4.5}$$

where

$$C = \sigma^2 I + \sum_{j=1}^{m}\tau_j x_j x_j^T \tag{4.6}$$

is the covariance matrix of y with a given τ.

4.4 Proximal Gradient Algorithm for Maximum A Posteriori Estimation

We focus on the $\hat{\tau}$, as the estimation of τ, while two unknown parameters μ and σ_0^2 can be obtained by setting $\frac{\partial L(\theta)}{\partial \mu} = 0$ and $\frac{\partial L(\theta)}{\partial \sigma_0^2} = 0$. Previously, we employed a coordinate ascent algorithm to infer a sparse τ from (4.5), which estimates one $\hat{\tau}_j$ at a time and is slow. In this chapter, we have developed a novel fast proximal gradient algorithm method to estimate unknown parameters τ.

Let

$$J(\tau) = f(\tau) + g(\tau), \tag{4.7}$$

where

$$g(\tau) = \lambda \sum_{j=1}^{m} \tau_j, \tag{4.8}$$

$$f(\tau) = \frac{1}{2}\left[\log|C| + \tilde{y}^T C^{-1}\tilde{y}\right], \tag{4.9}$$

and $\tilde{y} = (y - \mu)$. We then use proximal gradient algorithm to minimize $J(\tau)$. The proximal gradient method is an iterative algorithm with the $k+1$ step being:

$$\tau^{k+1} = prox_{\lambda^k g}\left([\tau^k - \lambda^k D^k \nabla f(\tau^k)]^+\right), \tag{4.10}$$

where $\nabla f(\tau^k) = \left[\frac{\partial f(\tau)}{\partial \tau_1}, \cdots, \frac{\partial f(\tau)}{\partial \tau_m}\right]^T_{\tau=\tau^k}$, $D^k = \left[diag\left(\frac{\partial^2 f(\tau)}{\partial \tau_1^2} ... \frac{\partial^2 f(\tau)}{\partial \tau_p^2}\right)_{\tau=\tau^k}\right]^{-1}$ and superscript k denotes the value obtained from k^{th} step.

Let $v^k = \left[\tilde{v}^k\right]^+ = \left[\tau^k - \lambda^k D^k \nabla f(\tau^k)\right]^+$. The proximate operator is defined as $prox_{\lambda^k g} := \arg\min_{\tau_k}\left(g(\tau) + \frac{1}{2\lambda^k}\|\tau - V^k\|_2^2\right)$ [PB13], where $\lambda_k > 0$ is a step size to be determined. Note that $v_j^k = \begin{cases} \tilde{v}_j^k, & \text{if } \tilde{v}_j^k > 0 \\ 0 & \text{o.w.} \end{cases}$, $j = 1, 2, \cdots, m$. Solving equation (4.10), we obtain:

$$\tau_j^{k+1} = \begin{cases} v_j^k - \lambda\lambda^k & \text{if } v_i^k > \lambda^k\lambda, \\ 0 & \text{o.w.} \end{cases} \tag{4.11}$$

We use the line search algorithm with the proximal gradient method to obtain τ^{k+1} and λ^k as shown in Table 4.1.

In Table 4.1, the calculation of v^k in each iteration requires evaluating derivative components $\frac{\partial f(\tau)}{\partial \tau_1}$ and $\frac{\partial^2 f(\tau)}{\partial \tau_1^2}$, while the exit criterion requires calculating $f(\tau)$. How-

ever, the calculation of $f(\tau)$ requires both $|C|$ and C^{-1}. Now, we derive more efficient expressions for these.

Given the equation (4.6), we can find an alternative form for C as

$$C = C_{-i} + \tau_i x_i x_i^T, \tag{4.12}$$

where $C_{-i} = \sigma^2 I + \sum_{j=1, j\neq i}^{p} \tau_j x_j x_j^T$. This facilitates finding an expression for modulus of C given by

$$\begin{aligned} |C| &= |C_{-i}(I + \tau_i C_{-i}^{-1} x_i x_i^T)| \\ &= |C_{-i}||I + \tau_i C_{-i}^{-1} x_i x_i^T| \\ &= |C_{-i}|(1 + \tau_i x_i^T C_{-i}^{-1} x_i). \end{aligned} \tag{4.13}$$

However, since C is a relatively large matrix of size $n \times n$, direct calculation of C_{-i} results in large computational complexity. To alleviate this problem, we use the Woodbury identity to find a less expensive alternative for this as:

$$C^{-1} = C_{-i}^{-1} - \frac{c_{-i}^{-1} x_i x_i^T C_{-i}^{-1}}{1/\tau_i + x_i^T C_{-i}^{-1} x_i}. \tag{4.14}$$

By combining these, we gain a more concise expression for $f(\tau)$ (4.9) as:

$$\begin{aligned} f(\tau) &= \frac{1}{2}[\log |c_{-i} + \tilde{y}^T C_{-i}^{-1} \tilde{y} + \log(1 + \tau_i x_i^T C_{-i}^{-1} x_i)] - \frac{\tilde{y}^T C_{-i}^{-1} x_i^T x_i^T C_{-i}^{-1} y}{1/\tau_i + x_i^T C_{-i}^{-1} x_i} \\ &= f(\tau_{-i}) + f(\tau_i). \end{aligned} \tag{4.15}$$

Motivation for this alternate form of $f(\tau)$ will be more clear when we derive expressions for derivative components of $f(\tau)$. For simplification, we define q_i and s_i as:

$$\begin{aligned} q_i &\triangleq x_i^T C_{-i}^{-1} \tilde{y} \text{ and} \\ s_i &\triangleq x_i^T C_{-i}^{-1} x_i. \end{aligned} \tag{4.16}$$

This results in a more simplified form for $f(\tau_i)$ given by

$$f(\tau_i) = \frac{1}{2}\left[\log(1+\tau_i si) - \frac{q_i^2}{\frac{1}{\tau_i}+s_i}\right]. \tag{4.17}$$

With these expressions, it is not difficult to find the derivative components of $f(\tau)$ as

$$\begin{aligned}\frac{\partial f(\tau)}{\partial \tau_i} &= \frac{1}{2}[\frac{s_i}{1+\tau_i s_i} - \frac{q_i^2}{(1+\tau_i s_i)^2}] \\ &= \frac{1}{2}\frac{s_i - q_i^2 + s_i^2\tau_i}{(1+\tau_i s_i)^2}\end{aligned} \tag{4.18}$$

and

$$\frac{\partial^2 f(\tau)}{\partial \tau_i^2} = \frac{1}{2}[-\frac{s_i^2}{(1+\tau_i s_i)^2} + \frac{2q_i^2 s_i}{(1+\tau_i s_i)^3}]. \tag{4.19}$$

To further simplify the expressions, we define S_i and Q_i as:

$$\begin{aligned}x_i^T C^{-1}\tilde{y} &= q_i - \frac{s_i q_i}{1/\tau_i + s_i} = \frac{q_i}{1+\tau_i s_i} \triangleq Q_i \text{ and} \\ x_i^T C^{-1} x_i &= s_i - \frac{s_i q_i}{1/\tau_i + s_i} = \frac{s_i}{1+\tau_i s_i} \triangleq S_i.\end{aligned} \tag{4.20}$$

With these, the derivative components get more compacted forms given by

$$\frac{\partial f(\tau)}{\partial \tau_i} = \frac{1}{2}[x_i^T C^{-1} x_i - (x_i^T C^{-1}\tilde{y})^2] = \frac{1}{2}[S_i - Q_i^2] \tag{4.21}$$

and

$$\begin{aligned}\frac{\partial^2 f(\tau)}{\partial \tau_i^2} &= \frac{1}{2}[-(x_i^T C^{-1} x_i)^2 + 2(x_i^T C^{-1} y)^2 x_i^T C^{-1} x_i] \\ &= \frac{1}{2}[-S_i^2 + 2Q_i^2 S_i].\end{aligned} \tag{4.22}$$

The major computational complexity is in calculating Q_i, S_i and $f(\tau)$, because calculation of these requires an inversion of matrix C. However, since C is a relatively

large matrix of size $n \times n$, direct calculation of C^{-1} has a computational complexity of $O(n^3)$. When n is large, this dramatically affects the performance. To avoid this burden, we derive a solution that results in a much lower computational complexity using the Woodbury matrix identity.

Let T be a diagonal matrix containing k non zero τ_js and $\tilde{X}$ be $n \times k$ matrix containing corresponding x_js. This gives an alternate form for C as:

$$C = \sigma^2 I + \tilde{X}^T \tilde{X}^T. \tag{4.23}$$

Now, using the Woodbury identity, we can find the inverse of C as:

$$C^{-1} = \sigma^{-2} I - \sigma^{-4} \tilde{X} \Sigma \tilde{X}^T, \tag{4.24}$$

where $\Sigma = (A + \sigma^{-2} \tilde{X}^T \tilde{X})^{-1}$ and $A = diag(\tilde{\tau}_1^{-1} ... \tilde{\tau}_k^{-1})$. As one can see, this has a computational complexity of $O(k^3)$. Note that, due to the sparsity of the model, $k \ll n$, that leads to a much lower computational complexity than its original form.

In the same direction, we can find expressions for Q_i and S_i with much lower computational complexity given by

$$\begin{aligned} Q_i &= \sigma^{-2} x_i^T \tilde{y} - \sigma^{-4} x_i^T \tilde{X} \Sigma \tilde{X}^T \tilde{y} \text{ and} \\ S_i &= \sigma^{-2} x_i^T x_i - \sigma^{-4} x_i^T \tilde{X} \Sigma \tilde{X}^T x_i. \end{aligned} \tag{4.25}$$

With these, it is not difficult to find an alternative expression for $f(\tau)$ with a lower computational complexity than its original complexity of $O(n^3)$ as follows:

$$\begin{aligned} f(\tau) &= \log|C| + \tilde{y}^T C^{-1}\tilde{y} \\ &= |\sigma^2 I + \tilde{X}^T\tilde{X}^T| + \tilde{y}^T C^{-1} \\ &= (\sigma^2)^n |I + \sigma^{-2}\tilde{X}^T\tilde{X}^T| + \tilde{y}^T C^{-1} \\ &= (\sigma_2)^n |I_{k\times k} + \sigma^{-2}\tilde{X}^T\tilde{X}^T| + \tilde{y}^T C^{-1} \\ &= (\sigma^2)^{n-k} |\sigma^2 I_{k\times k} + \tilde{X}^T\tilde{X}^T| + \tilde{y}^T C^{-1} \\ &= (\sigma^2)^{n-k} |\sigma^2 I_{k\times k} + \tilde{X}^T\tilde{X}^T| + \sigma^{-2}||\tilde{y}||^2 - \sigma^{-4}\tilde{y}^T\tilde{X}\Sigma\tilde{X}^T\tilde{y}. \end{aligned} \tag{4.26}$$

As can be seen from the final expression, this has a computational complexity of $max(\{O(k^3), O(nk)\})$, which is much lower than its original form of $O(n^3)$ due to the sparsity of the model.

Other two unknown parameters μ and σ_0^2 can be obtained by letting $\frac{\partial L(\theta)}{\partial \mu} = 0$ and $\frac{\partial L(\theta)}{\partial \sigma_0^2} = 0$. This results in

$$\mu^{k+1} = \frac{1^T C^{-1} y}{1^T C^{-1} 1}, \tag{4.27}$$

where 1 is a vector whose elements are all 1, and

$$\sigma^{2(k+1)} = \frac{||\tilde{y} - \tilde{X}u||^2}{n - k + \sum_{i=1}^{k} \frac{[\Sigma]_{ii}}{\tilde{\tau}_i}}, \tag{4.28}$$

where Σ_{ii} is the i^{th} diagonal element of Σ and

$$u = \sigma^{2(k)}\Sigma\tilde{X}^T\tilde{y}. \tag{4.29}$$

After these parameters are estimated, the posterior distribution of β can be found. Specifically, the proximal gradient algorithm will select m' (typically $m' \ll m$)

nonzero elements of β corresponding to the nonzero elements in $\hat{\tau}$, which is denoted as a $m' \times 1$ vector β', that corresponds to finite $\hat{\tau}_j$s. Let $\hat{\tau}$ be a $m' \times 1$ vector contain all finite τ_js. Given $\hat{\tau}$, it is not difficult to show that the posterior distribution of β' is a Gaussian distribution with mean $\hat{\beta}' = \sigma_0^2 \hat{\Sigma} \tilde{X}(y - \mu)$ and covariance $\hat{\Sigma} = (A + \sigma_0^{-2} \tilde{X}^T \tilde{X})^{-1}$, where $\tilde{X}$ is an $n \times m'$ matrix that contains the columns of X corresponding to β', and A is a diagonal matrix with $\frac{1}{\hat{\tau}_1}, \cdots, \frac{1}{\hat{\tau}_{m'}}$ on its diagonal.

For the final algorithm, we employ a two level iterative technique to evaluate the model. In the inner loop, we assume that the μ and σ_0^2 are known and fixed, then we estimate the τ^{k+1} and λ^k. In the outer loop, based on estimations from the inner loop, we refine μ and σ_0^2 and check for convergence. The convergence of the outer loop is achieved when both of the following conditions are met.

1. Indices of elements of $\tilde{\tau}$ do not change; note that $\tilde{\tau}$ contains the nonzero elements of τ which is a long vector of $m \times 1$, but length of $\tilde{\tau}$ should be relatively small.

2. $\frac{|\tilde{\tau}^k - \tilde{\tau}^{k-1}|^2}{|\tilde{\tau}^{k-1}|^2} < \epsilon$, where ϵ is a small number (e.g., $1e^{-3}$ or $1e^{-4}$)

Having defined all necessary expressions, we summarize our EBlasso algorithm in Table 4.4.

4.5 Experiment Procedure

The population of an F2 family derived from the cross of two inbred lines with $m = 481$ genetic markers, which were evenly spaced on a large chromosome of 2400 cM (interval $d = 5$ cM), was simulated. For the three genotypes, A_1A_1, A_1A_2 and

Table 4.1: Proximal method with line search algorithm

Begin
Let $\alpha = \lambda^{k-1}, \gamma \in (0,1)$
$\hat{f}_\alpha(z, v^k) \triangleq f(v^k) + \nabla f(v^k)^T(z - v^k) + \frac{1}{2\alpha}\|z - v^k\|_2^2$
while true **do**
$v^k = \left[\tilde{v}^k\right]^+ = \left[\tau^k - \lambda^k D^k \nabla f(\tau^k)\right]^+$
$z = prox_{\alpha g}(v^k)$
if $f(\tau) \leq \hat{f}_\alpha(z, v^k)$ **then**
break
end if
$\alpha = \gamma\alpha$
end while
$\lambda^k = \alpha$, $\tau^{k+1} = z$
End

Table 4.2: EBlaso with proximal gradient

Begin
Initialization: Choose $\lambda > 0$, $u = 1^T \frac{y}{n}$, $\tilde{y} = y - u1$, $\gamma = 0.5$, $\lambda^0 = 1$, $\sigma^2 = \frac{0.1||\tilde{y}||^2}{n}$
repeat
Begin
SubRoutine: $(\tilde{\tau}^{k+1}, \Sigma, \tilde{X}) = proxm(\tau^k, \lambda^{k-1}, \gamma)$ (4.1)
End
Calculate μ^{k+1}
Calculate $\sigma^{2(k+1)}$
until Indices of elements of $\tilde{\tau}$ do not change AND $\frac{|\tilde{\tau}^k - \tilde{\tau}^{k-1}|^2}{|\tilde{\tau}^{k-1}|^2} < \epsilon$, where ϵ is a small number
End

A_2A_2 of individual i at marker j, the dummy variable was defined as $x_{ij} = 1, 0, -1$, respectively. The assumption was that the QTLs coincided with markers. If they were not on markers, they may still have been detected, as the correlation between a QTL and a nearby marker was high. However, to get the same power of detection a slightly larger sample size may be required.

We performed data simulations based on the F2 population with 20 main QTL effects, whose effect sizes were randomly generated from a normal distribution with mean zero and variance equals to four. Environmental effects were not simulated. The true population mean was $\mu = 100$ and the residual variance was $\sigma_0^2 = 10$.

We did two phases of simulations. In the first phase, we generated a data set with sample size 1000. This was used to evaluate the power of detection (PD) and the false discovery rate (FDR) of our EBLasso proximal gradient algorithm over the coordinate ascent algorithm [HXC13]. We used the prediction error (PE) [Tib96] obtained from a ten-fold cross validation process to select the value of the hyperparameter λ for both algorithms. For estimated phenotype value $\hat{y}_i = \hat{\mu} + \tilde{x}_i u$, the PE was calculated by:

$$PE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2. \tag{4.30}$$

The results of this phase primarily illustrate parameter estimation and cross validation. In the second phase, for a measure of the robustness of our algorithm, we generated 100 replicates with sample sizes 200, 400, 600, 800 and 1000. Computational time, PD and FDR were evaluated to have a better comparison with the coordinate ascent algorithm.

To have a fair comparison, both our algorithm and the comparison algorithm were implemented in C/C++ following the same level of coding and algorithmic techniques. The data were analyzed on a personal computer (PC) with Intel Xeon 2.93GHz CPU running 64 bit Windows server 2008.

4.6 Results and Discussion

4.6.1 Phase 1

We fixed the λ_k and γ parameters in our algorithm at 1.0 and 0.5, respectively, and used the ten-fold cross validation to estimate the hyperparameter λ. We could also have used the same cross validation technique to evaluate λ_k and γ as well, but that requires more time in the training phase. However, as we observed in the simulation study, this did not adversely affect the results.

Even though cross validation allows us to pick the optimum λ value, it is challenging to find the set of minimum candidate values for the λ to avoid extensive computational overhead. From a brute force perspective, we could have used very large sets of values spaced evenly in a linear scale, but that would dramatically increase the training time. Especially when dealing with a large number of data sets, this would not even be feasible. To overcome this problem, we used an evenly spaced log scale for candidate values for λ. Furthermore, we formulated a method to infer the lower and upper bound of λs from the data set. This will play a vital role in dealing with many data sets in Phase2 that drastically expedites the training phase.

Specifically, we defined the upper bound of λ as:

$$\lambda_{max} = \frac{1}{N} max(xj * (y - \mu)). \tag{4.31}$$

The lower bound of λ, λ_{min}, was calculated by $\lambda_{max} \times 0.001$. We used the full data set in evaluating the bounds since we needed to use the same sets of candidate λs with all the ten-fold cross validation data sets. To obtain the candidate values, we equally divided the range $log(\lambda_{max} - \lambda_{min})$ into 40 steps, which resulted in 41 candidate values. Figure 4.1 demonstrates the variation of PE with both linear and log scales.

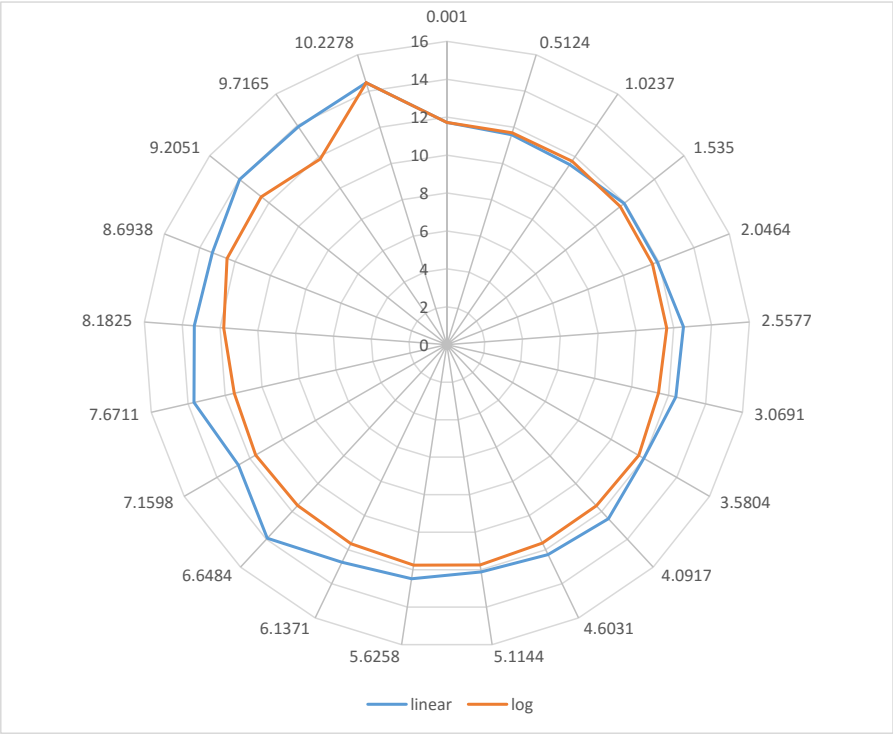

Figure 4.1: Prediction error changes based on the log and linear scale of λ.

In general, the values close to the lower and upper bounds get larger prediction errors, and this is clearly depicted in Figure 4.1. Furthermore, the log scale keeps more candidate points in the interesting region, still giving a good visibility on bounds.

It is interesting to observe the impact of the cross validation results over the actual detections. To examine this, we tabled a few candidate values used in ten-fold cross validations with their true positive (TP) and false positive (FP) detections, as shown in Table 4.3.

The average PE and the standard error were obtained from ten-fold cross validation. The number of effects was obtained by using all 1000 samples not from cross validation samples. All the effects counted had a *p-value* ≤ 0.01. According to the Table 4.3, we observed a lowest prediction error 11.4984, which resulted in 17 true detections and a single false detection.

Our algorithm considers all variables simultaneously. If you recall the algorithm, in each inner iteration, we added non zero τ_is to the model when $v_i^k - \lambda^k \lambda > 0$. We observed that, adding a small positive number ϵ to the right hand side will greatly reduce the number of iterations required for convergence. The reason is that it takes some additional iterations to get rid of those less impactful τ_is. For this reason, we used $\epsilon = 0.001$ with all our simulations.

As shown in Table 4.4, our algorithm consists of two loops, the inner loop and the outer loop. Basically, we can improve performance further by filtering out less interesting parameters just before we transition from the outer loop to the inner loop. We used a t-test with *p-value* < 0.01 to filter out some noisy parameters. A similar technique was used by Yi and Banerjee in [YB09].

Table 4.3: Variaton of detections over λ with prediction error obtained from ten-fold cross validation.

λ	True Detections	False Detections	PE ± STD
0.1023	15	2	11.7435 ± 0.7124
0.1288	15	2	11.6724 ± 0.8342
0.1621	16	0	11.759 ± 0.7231
0.2041	15	0	11.6874± 0.7568
0.2569	14	2	11.6539± 0.8245
0.3234	17	4	11.5041± 0.8561
0.4072	15	1	11.5722 ± 0.8345
0.5126	15	2	11.5913± 0.7634
0.6453	16	2	11.6489 ± 0.7823
0.8124	15	1	11.6662 ± 0.7458
1.0228	17	1	11.4984 ± 0.7452
1.2876	16	1	11.7545 ± 0.8345
1.621	17	2	11.8032 ± 0.8256
2.0407	17	1	11.9084 ± 0.8235
2.5691	16	1	12.456 ± 0.7812
3.2343	15	0	12.0436 ± 0.7954
4.0718	12	2	12.5187 ± 0.9674
5.1261	13	1	12.1537 ± 0.9345
6.4533	16	2	11.8593 ± 0.9567
8.1243	15	1	13.1843 ± 0.8758
10.2278	14	0	14.4569 ± 0.9124

We used the same ten-fold cross validation technique for both algorithms to find the optimum value for λ. Once the algorithms were completed, each algorithm output some non-zero values for the corresponding indexes of x_i. For those x_is not in the model, we can declare that they do not affect the quantitative trait because their regression coefficient is zero. For those x_is in the matrix $\tilde{X}$, the posterior distribution of their regression coefficients $\tilde{\beta}$ is Gaussian with covariance Σ and mean u. We then used *t-statistics* to test if $\tilde{\beta} \neq 0$ with *p-value* $= 0.01$. Table 5.4 summarizes the final detection results for both algorithms based on optimum λ values.

As one can see from Table 5.4, the proximal gradient algorithm detected 17 TPs, while the coordinate accent algorithm detected only 15. The proximal gradient algorithm detected the marker 466 as a false positive, while the coordinate ascent algorithm detected the marker 78 as a false positive. Based on CPU times, our algorithm accelerated the EBlasso method by 73.04% ($\frac{-(t_{ProximalGradient} - t_{CoordinateAscent})}{t_{CoordinateAscent}} \times 100\%$).

4.6.2 Phase 2

In this phase, to see if the proximal gradient algorithm could estimate QTL effects robustly, we replicated 100 data sets with sample sizes 200, 400, 600, 800, and 1000 using the same procedures described in Phase 1. Primary focus here is to evaluate the consistency of the results and measure the impact on the detection rate over different sample sizes. We used the same technique we detailed in Phase 1 for the ten-fold cross validation. The PD and FDR were calculated as an average of 100 replicated

Table 4.4: True estimated QTL effects for the simulated data with main effects.

Markers	Position (cM)	Proximal gradient (Is detected?)	Coordinate ascent (Is detected?)
11	50	Yes	Yes
26	125	Yes	Yes
42	205	Yes	Yes
48	235	Yes	Yes
72	355	No	No
73	360	Yes	Yes
123	610	Yes	Yes
127	630	Yes	Yes
161	800	Yes	Yes
181	900	Yes	No
182	905	Yes	Yes
185	920	Yes	No
221	1100	Yes	Yes
243	1210	Yes	Yes
262	1305	Yes	Yes
268	1335	No	No
270	1345	No	No
274	1365	Yes	Yes
361	1800	Yes	Yes
461	2300	Yes	Yes

simulations. Figure 4.2 and Figure 4.3 show the PD and the FDR plots for both algorithms, respectively.

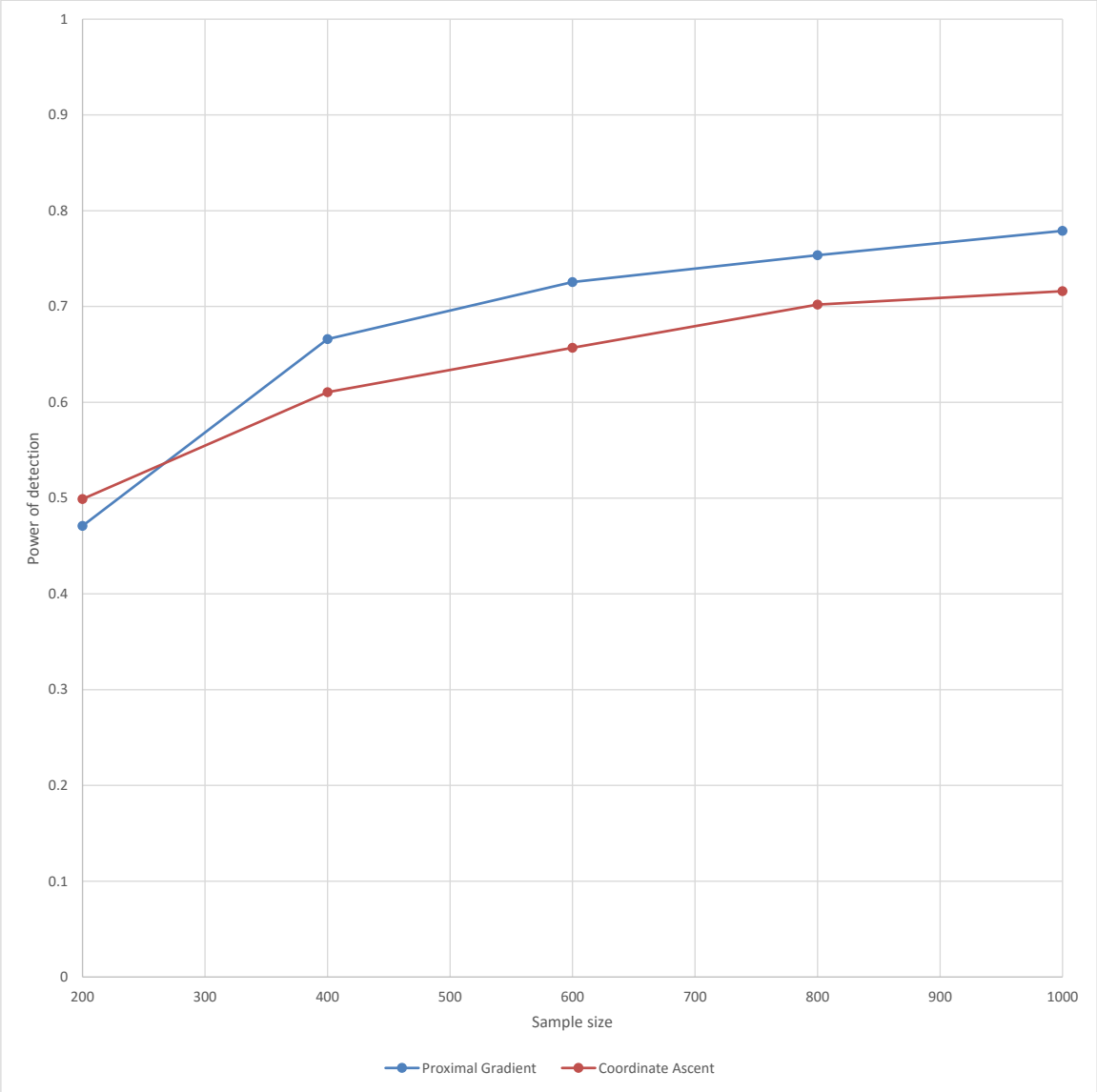

Figure 4.2: Power of detection for the proximal gradient and coordinate ascent algorithm. Performance data were obtained from mean of 100 replicas for different sample sizes (n = 200, 400, 600, 800, 1000).

As one can see from the results, the proximal gradient offered better PD and FDR over the coordinate ascent algorithm. However, the major contribution of our work is to the performance aspect of the EBlasso method. We calculated the average CPU

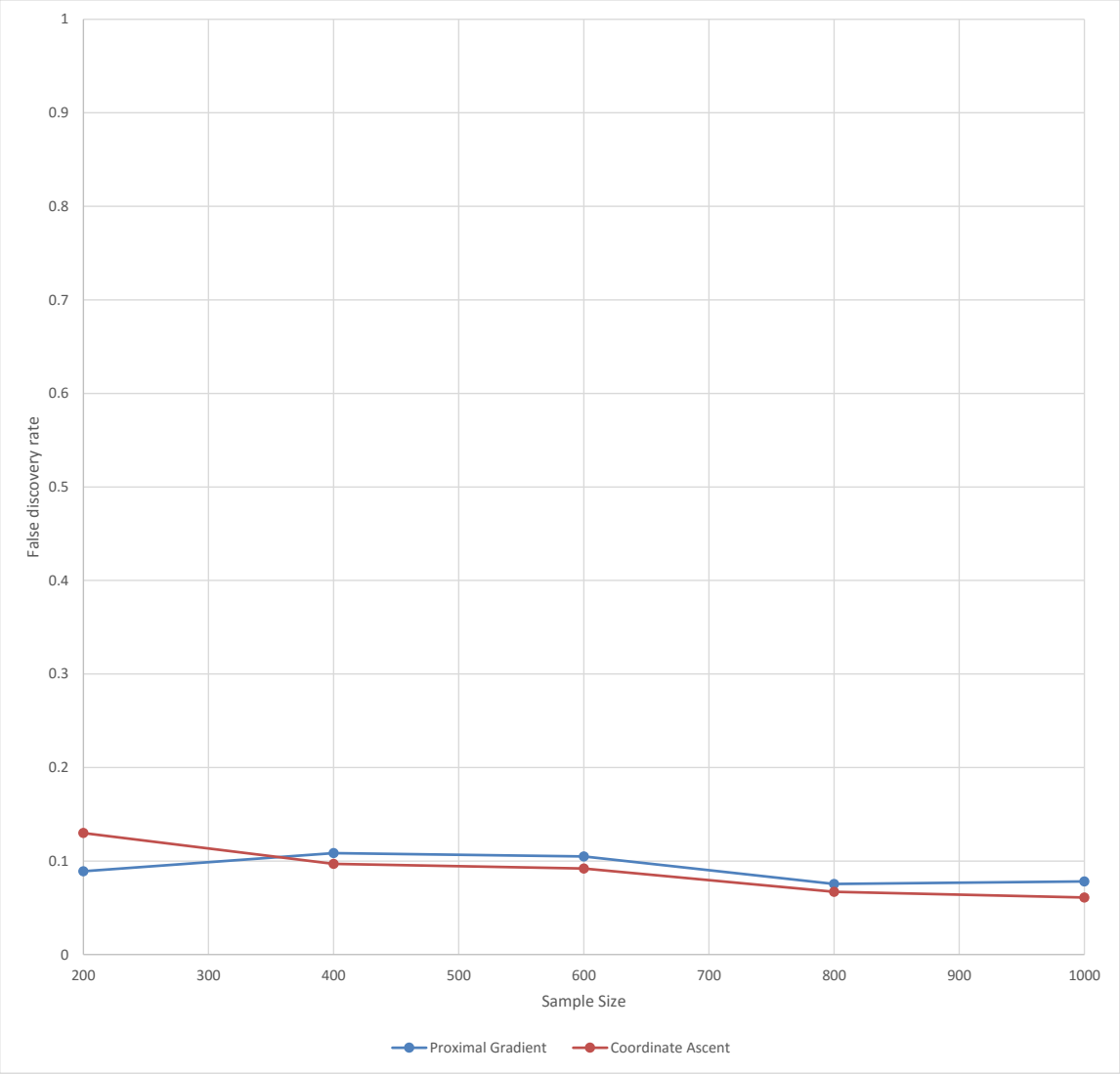

Figure 4.3: False discovery rate for the proximal gradient and coordinate ascent algorithm. Performance data were obtained from mean of 100 replicas for different sample sizes ($n = 200, 400, 600, 800, 1000$).

time for both algorithms for the same 100 data sets used in the previous step. The results are plotted in Figure 4.4. As one can see, our new proximal gradient algorithm accelerates the EBlasso method by 1.5 - 2.5 times based on 100 replicates with a range of different sample sizes.

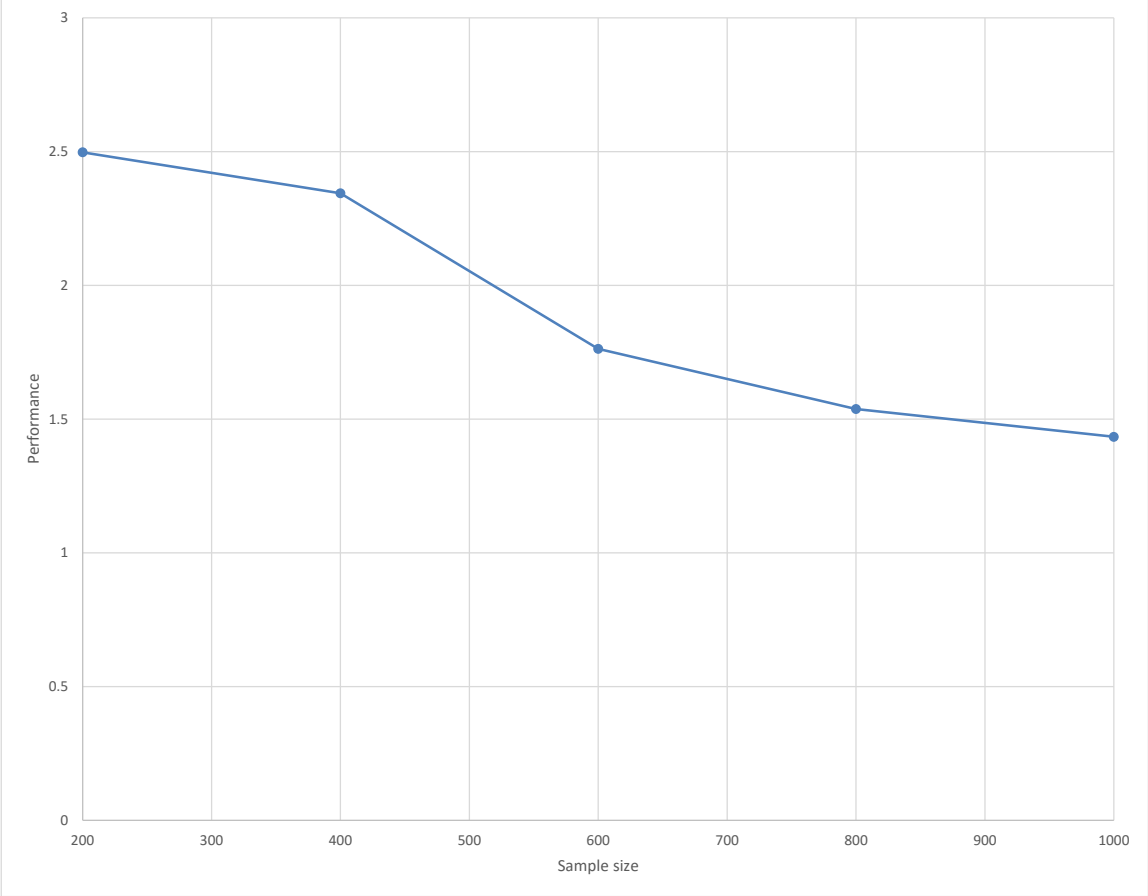

Figure 4.4: Performance is calculated as $t_{CoordinateAscent}/t_{ProximalGradient}$. Performance data were obtained by using the mean of 100 replicates.

The simulation result demonstrated that the better optimization approach of the proximal gradient algorithm improves EBlasso from both computational efficiency and detection accuracy. Specifically, in the proximal gradient method, the nonzero effects in the model are considered simultaneously, and the global optimum is obtained

faster and more accurately compared with the greedy approach that considers one effect at a time in the coordinate ascent algorithm.

Since our proximal gradient algorithm calculates all the variables at the same time instead of one by one as in coordinate ascent, there is more room for improvement by parallelizing some of the calculations. We did not use any parallel computations for the results presented here meaning that there is more room for performance improvements.

4.7 Summary

Leveraging on recent advancements of proximal algorithms and their applications in Lasso type of problems, we developed a novel proximal gradient algorithm [AHC14] for the EBlasso method. Our EBlasso method is based on a Bayesian linear regression model that uses the same two-level hierarchical prior for the regression coefficients as the one used in the Bayesian Lasso linear regression model [YX08, PC08, CHX11]. It first estimates the variance of the regression coefficients and then finds the posterior distribution of the regression coefficients based on the estimated variance.

Compared to the previous implementation with the greedy coordinate ascent algorithm that adds, removes or re-estimates one variable from the model at a time, the proximal gradient algorithm considers all variables simultaneously and maps points towards the minimum of the objective function in (4.5). In the proximal algorithm, the base operation is to evaluate the proximal operator, which is typically a specialized method with closed form solutions [PB13].

Based on the simulation study by Cai et al. [CHX11], our previous algorithm offers better performance than the other state-of-the-art methods such as Lasso [Tib96] and elastic net [ZH05], in terms of PD, FDR. Thanks to the high level of abstraction with great flexibility, proximal algorithms and their particular implementations have been shown to be efficient at solving regularized regression such as Lasso [BT09] and structured input and output Lasso problem [LX12]. With the objective function evaluated efficiently, our simulation results show that the new method provides better accuracy and faster computational speed in multiple QTL mapping [AHC14].

Our recent studies demonstrate that EBlasso has a broad range of applications, such as whole genome QTL mapping and pathway-based genome-wide association study (GWAS) [HXC14b, HMVC14]. When the number of possible effects is very large in QTL models with both main and epistatic effects, the computation time becomes a critical concern. Given that high density marker maps can be easily obtained due to advancements in sequencing technology, it would be worthwhile to explore these areas with the new method developed in this chapter.

The advancement in developing proximal algorithms for the EBlasso also presents future research avenues. For example, both EBlasso-NEG [CHX11, HXC13] and our recent developed EBEN [HXC14a] have more parameters and require much more computation in cross validation to identify their optimal values. It would be very useful to have more efficient proximal algorithms for these methods.

CHAPTER 5

Empirical Bayesian Lasso Proximal Gradient Algorithm with Normal, Exponential and Gamma Hierarchical Prior Distributions for Fast Learning

In the previous chapter, we developed a novel proximal gradient optimization algorithm for the empirical Bayesian Lasso (EBlasso) NE model that resulted in a better power of detection (PD), false discovery rate (FDR) and greatly increased performance. Continuing in the same direction, we develop here a novel proximal gradient based EBlasso algorithm with Normal, Exponential and Gamma (NEG) prior distributions. This type of hierarchical prior distribution results in a fast convergence while alleviating the problem of choosing an inappropriate value for λ. The key to the increased performance is that it considers all the model parameters simultaneously. Our new algorithm, running on a personal computer, could easily handle a linear QTL model with more than 100,000 variables with extremely fast convergence. To further enhance the detection rate, we develop a novel hybrid algorithm that combines

the fast convergences of our proximal gradient method with a simplified secondary algorithm to achieve higher level of QTL detections, while keeping the false positive effects at a much lower level.

5.1 Background

Analysis of quantitative trait loci (QTLs) is a statistical method that links two types of information, phenotypic data (trait measurements) and genotypic data (usually molecular markers), in an attempt to explain the genetic basis of variation in complex traits [FM96, Kea98, LW98]. QTL analysis allows researchers in fields as diverse as agriculture and medicine to link certain complex phenotypes to specific regions of chromosomes. The goal of this process is to identify the action, interaction, number, and precise location of these regions.

It is highly desirable to analyze a large number of loci simultaneously due to the physical linkage of epistatic interactions among multiple QTLs. Since hundreds of thousands of genomic loci or markers are usually genotyped and involved in QTL mapping studies, including all these markers and their possible interactions in a model leads to a huge number of model variables, typically much larger than the sample size. This is not only entails huge computation that is not affordable to existing QTL mapping methods, but also may reduce power of detection and/or increase false discovery rate.

There are two general techniques often employed to handle such oversaturated models: variable selection and shrinkage. Variable selection attempts to identify

a subset of all possible generic effects that best explain the phenotypic variation, typically using a stepwise search procedure in conjunction with a selection criterion such as the Bayesian information criterion (BIC) [Sch78]. On the other hand, a shrinkage method includes all variables in the model, but uses a penalty function of the variables or appropriate prior distributions on the variables to shrink less effective variables toward zero. Ridge regression [HK70] and the least absolute shrinkage and selection operator (LASSO) [Tib96] are examples of early shrinkage methods. However, the Bayesian shrinkage method [OS09] has received considerable attention recently.

Bayesian shrinkage has been applied to multiple QTL mappings [Xu03, WYL$^+$05, HS06, HETZ07, YX08]. Basically, Markov chain Monte Carlo (MCMC) simulation is used with all of these works to fit the Bayesian model and provide comprehensive information about the model drawing from the posterior distribution of the model variables. However, despite the advances in the development of the MCMC simulation algorithms [RC04], MCMC simulations are heavily computationally intensive and time consuming, as we saw in Chapter 2. To reduce the computational burden of the fully Bayesian approach relying on MCMC simulations, Xu [Xu07] developed an empirical Bayes (EB) method that uses a properly chosen prior distribution for the model variables to shrink variables toward zero.

Xu [Xu07] demonstrated that the EB method can handle a large number of model variables simultaneously. More recently, the EB method was extended to handle classification predictor variables [Xu10]. The EB method estimates the variance components utilizing numerical algorithms such as the simplex algorithm [NM65]. Al-

though it requires much less computational power over the fully Bayesian approach, the efficiency is limited by the numerical optimization algorithm. On the other hand, the machine learning community developed a very efficient EB method, namely, the relevance vector machine (RVM) [Tip01, TF03].

The RVM has increased speed because it estimates the variance components in a closed form. Furthermore, it employs a few other algorithmic techniques in the implementation to make it faster. It assumes a uniform prior distribution for the variance components. Although this choice of the prior distribution gets rid of any hyperparameters to be pre-specified, it lacks the flexibility of adjusting the degree of shrinkage needed for analyzing a specific data set. Particularly, its uniform prior distribution may not provide enough shrinkage in multiple QTL mapping, which includes a very large number of possible effects, often resulting in a large number of false effects [Xu10].

We developed an efficient empirical Bayesian Lasso (EBlasso) algorithm based on the Bayesian LASSO model [YX08, PC08] with an exponential prior distribution for the variance components in contrast to the inverse χ^2 distribution for the variance components used by the EB method [Xu07]. Based on the simulation study in [CHX11], the EBlasso method demonstrated outstanding performance in terms of detection rate, false discovery rate and speed. In fact, the EBlasso method was orders of magnitude faster than the EB method. However, the greedy coordinate ascent algorithm used in the EBlasso method considers one effect at a time in an iterative fashion [CHX11]. This makes the algorithm comparatively slow when dealing with a large number of model variables, even though it is considered much faster than the

EB method. Since the model grows one effect at a time, it also fails to capture the complex relations across multiple effects simultaneously.

Capitalizing on this idea, we have developed a novel proximal gradient optimization algorithm for the EBlasso method. Our model uses three-level hierarchical prior distribution, normal, exponential and gamma, for the variance components. Instead of the previous implementation with the greedy coordinate ascent algorithm that adds, removes or reestimates one variable from the model at a time, the proximal gradient algorithm considers all variables simultaneously and maps points toward the minimum of the objective function. Thanks to the advanced optimization algorithm and other algorithmic techniques used, the new proximal gradient based algorithm is capable of providing better detection rate, lower false discovery rate and superior speed over the coordinate ascent algorithm. Furthermore, it is capable of handling extremely large numbers of parameters. As simulation results indicate, the new proximal gradient algorithm is orders of magnitude faster than the coordinate ascent algorithm. We also develop a novel hybrid model that is capable of detecting more QTL effects than its vanilla flavor (regular proximal gradient method). Moreover, real data analysis demonstrates that the proximal gradient algorithm is able to detect more QTL effects with impressive speed over the coordinate ascent algorithm.

5.2 Bayesian Multiple Linear Regression Model for QTLs

Capitalizing on the concepts from the previous chapter, we utilize a Bayesian multiple linear regression model to infer genotype and quantitative phenotype associations. We also consider environmental effects, main and epistatic effects of all markers and gene-environment ($G \times E$) interactions. Let y_i be the phenotypic value of a quantitative trait of the i^{th} individual in a mapping population. Suppose we observe y_i, $i = 1, \cdots, n$ of n individuals and collect them into a vector $y = [y_1, y_2, \cdots, y_n]^T$. In these n individuals, suppose p environmental covariates are observed and q genetic markers genotyped. Let covariate l and genotype of marker j of individual i be x_{Eil} and x_{Gij}, respectively. Let us define $X_{Ei} = [x_{Ei1}, x_{Ei2}, \cdots, x_{Eip}]^T$ and $X_{Gi} = [x_{Gi1}, x_{Gi2}, \cdots, x_{Giq}]^T$. The interaction between any two effects is modeled as the element-wise product of the corresponding main effects. Let X_{GGi} be a $\frac{1}{2}q(q-1) \times 1$ vector containing x_{Gij}. $x_{Gij'}$, $j = 1, \cdots, q-1$, $j' > j$, X_{GEi} be a $pq \times 1$ vector containing x_{Eil}. x_{Gij}, $l = 1, \cdots, p$, $j = 1, \cdots, q$. Then we have the following linear regression model for y:

$$y = \mu + X_E\beta_E + X_G\beta_G + X_{GG}\beta_{GG} + X_{GE}\beta_{GE} + e, \tag{5.1}$$

where μ is the population mean, vectors β_E and β_G represent the environmental effects and the main effects of all markers, respectively, and vectors β_{GG} and β_{GE} capture the epistatic effects and the gene-environment interactions, respectively. $X_E = [X_{E1}, X_{E2}, \cdots, X_{En}]^T$, $X_G = [X_{G1}, X_{G2}, \cdots, X_{Gn}]^T$, $X_{GG} = [X_{GG1}, X_{GG2}, \cdots, X_{GGn}]^T$,

$X_{GE} = [X_{GE1}, X_{GE2}, \cdots, X_{GEn}]^T$ are the corresponding design matrices of different effects, and e is the residual error that follows a normal distribution with zero-mean and covariance $\sigma_0^2 I$.

The design matrix X_G depends on a specific genetic model. We adopt the same Cockerham genetic model we used in Chapter 4. We can write y in more compact form as:

$$y = \mu + X\beta + e, \tag{5.2}$$

where $\beta = [\beta_0, \beta_E^T, \beta_G^T, \beta_{GG}^T, \beta_{GE}^T]^T$, $X = [X_E, X_E G, X_{GG}, X_{GE}]$. Let environmental covariates be p and the number of markers whose main effects are additive be q. This makes the size of the matrix X as $n \times m$ where $m = p + \frac{1}{2}q(q+1)$. Typically, $m \gg n$. Furthermore, m is even larger if dominance effects of the markers are considered. Our goal is to estimate all possible environmental and genetic effects on y manifested in the regression coefficients β, which is a challenging problem because $m \gg n$. However, we would expect that most elements of β to be zeros, and thus we have a sparse linear model. We will adopt the Blasso model [PC08, YX08] where appropriate prior distributions are assigned to the elements of β as described in the next section.

5.3 NEG Hierarchical Prior Distribution

While complete model parameters of (5.2) are μ, σ_0^2 and β, let us first focus on β. We assign a non-informative uniform prior to μ and σ_0^2, $p(\mu) \propto 1$ and $p(\sigma_0^2) \propto 1$, respectively. We assume a three-level hierarchical model for β. Let us denote the elements of β as β_j, $j = 1, \cdots, m$. At the first level, β_j, $j = 1, \cdots, m$ follows an

independent normal distribution with zero mean and unknown variance τ_j : $\beta_j \sim N(0, \tau_j)$. At the second level, τ_j, $j = 1, 2, \cdots, m$, follows an independent exponential distribution with a common parameter λ: $p(\tau_j) = \lambda \exp(-\lambda \tau_j)$. For a given λ, the distribution of β_j is found to be the Laplace distribution: $p(\beta_j) = \sqrt{\frac{\lambda}{2}} \exp(-2\sqrt{\lambda}|\beta_j|)$, which is known to encourage the shrinkage of β toward zero [Tib96].

However, the degree of shrinkage strongly depends on the values of λ. To alleviate the problem of choosing an inappropriate value for λ, we add another level to the hierarchical model at which we assign a conjugate Gamma prior $Gamma(a, b)$ with a shape parameter $a > 0$ and an inverse scale parameter $b > 0$ to the parameter λ. As discussed in [YX08], we can pre-specify appropriate values for a and b so that the Gamma prior for λ is essentially non-informative. Let us define $\tau = [\tau_1, \tau_2, \cdots, \tau_m]^T$. Similar to the EB method of [Xu07], we first estimate parameter τ, σ_0^2 and μ then find the posterior distribution of β based on the estimated parameters. Since λ is the parameter we do not want to estimate, we can find the prior distribution of τ independent of λ as follows:

$$p(\tau) = \int_0^\infty p(\tau|\lambda)p(\lambda)d\lambda = \frac{a}{b(\tau/b + 1)^{a+1}}. \tag{5.3}$$

The posterior distribution of μ, β, τ and σ_0^2 is given by:

$$p(\mu, \beta, \tau, \sigma_0^2|y) \propto p(y|\mu, \sigma_0^2)p(\mu)p(\beta|\tau)p(\sigma_0^2). \tag{5.4}$$

We collect all parameters that need to be estimated as $\theta = (\mu, \sigma_0, \tau)$. The log marginal posterior distribution of θ can be found from (5.4) as:

$$L(\theta) = -\frac{1}{2}\left[\log|C| - \frac{1}{2}\tilde{y}^T C^{-1}\tilde{y}\right] - \sum_{j=1}^{m}(a+1)\log(\tau_i + b) + constant, \tag{5.5}$$

where $C = \sigma^2 I + \sum_{j=1}^{m} \tau_j x_j x_j^T$ is the covariance matrix of y with a given τ and $\tilde{y} = (y - \mu)$.

5.4 Proximal Gradient Approach

While two unknown parameters μ and σ_0^2 can be obtained by setting $\frac{\partial L(\theta)}{\partial \mu} = 0$ and $\frac{\partial L(\theta)}{\partial \sigma_0^2} = 0$, we will focus on the $\hat{\tau}$, the estimation of τ.

Let

$$J(\tau) = f(\tau) + g(\tau), \tag{5.6}$$

where $f(\tau) = \frac{1}{2}\left[\log|C| - \frac{1}{2}\tilde{y}^T C^{-1}\tilde{y}\right]$ and $g(\tau) = \sum_{j=1}^{m}(a+1)\log(\tau_i + b)$. Proximal method gives an estimate for τ, as $\hat{\tau} = \arg\min_\tau \left(g(\tau) + \frac{1}{2\lambda}||\tau - v||^2\right)$. This leads to an expression for $J(\tau_i)$ given by:

$$J(\tau_i) = \frac{1}{2\lambda}(\tau_i - v_i)^2 + (a+1)\log(\tau_i + b) \tag{5.7}$$

With this, we can derive an expression for the derivative of $J(\tau_i)$ with respect to τ_i as:

$$\begin{aligned} \frac{dJ(\tau_i)}{d\tau_i} &= \frac{1}{\lambda}(\tau_i - v_i) + \frac{a+1}{\tau_i + b} \\ &= \frac{(\tau_i - v_i)(\tau_i + b) + \lambda(a+1)}{\lambda(\tau_i + b)} \\ &= \frac{\tau_i^2 + (b - v_i)\tau_i - bv_i + \lambda(a+1)}{\lambda(\tau_i + b)}. \end{aligned} \tag{5.8}$$

Since the numerator of $\frac{dJ(\tau_i)}{d\tau_i}$, $\mathcal{N}(\tau_i)$, represents a quadratic equation of τ, the discriminant, Δ is given by:

$$\Delta \triangleq (b - v_i)^2 - 4[\lambda(a+1) - bv_i]. \tag{5.9}$$

Given this, let us consider possible solutions for τ, τ^*:

5.4.1 Case 1: $\Delta < 0$

Since τ_i is real, $\mathcal{N}(\tau_i) = 0$, means no solution. This implies that $\frac{dJ(\tau_i)}{d\tau_i} > 0$, that results in $\tau_i^* = 0$.

5.4.2 Case 2: $\Delta = 0$

This yields only a single solution and $\frac{dJ(\tau_i)}{d\tau_i} > 0$ at $\tau_i \neq \hat{\tau}_i$; thus it results in $\tau_i^* = 0$.

5.4.3 Case 3: $\Delta > 0$

In this case, $\mathcal{N}(\tau_i)$ has two solutions without constraint $\tau_i > 0$. Let us denote two roots of τ_i as r_{i1} and r_{i2}, given by:

$$\begin{aligned} r_{i1} &= \frac{1}{2}[v_i - b - \sqrt{\Delta}] \\ r_{i2} &= \frac{1}{2}[v_i - b + \sqrt{\Delta}] \end{aligned} \tag{5.10}$$

With these roots, let us consider three main scenarios:

a) $v_i - b < -\sqrt{\Delta}$

This makes $r_{i1}, r_{i2} < 0$, which implies $\tau_i^* = 0$. This helps in deriving boundaries of v_i as follows:

$$\begin{aligned} v_i - b &< 0 \\ (v_i - b)^2 &> \Delta \\ v_i^2 - 2bv_i + b^2 &> v_i^2 + b^2 + 2bv_i - 4\lambda(a+1) \\ v_i &< \frac{\lambda(a+1)}{b} \implies \\ v_i &< min\left(b, \frac{\lambda(a+1)}{b}\right) \end{aligned} \tag{5.11}$$

b) $v_i - b + \sqrt{\Delta} > 0$ **and** $v_i - b - \sqrt{\Delta} < 0$

This results in $r_{i1} < 0$ and $r_{i2} >$) giving $J'(\tau_i) := \begin{cases} < 0 & \text{if } \tau_i < r_{i2}, \\ & \\ > 0 & \text{if } \tau_i > r_{i2} \end{cases}$. This gives $\tau_i^* = r_{i2}$ and the boundaries of v_i given by:

$$\begin{aligned} |v_i - b| &< \sqrt{\Delta} \\ (v_i - b)^2 &< \Delta \implies \\ v_i &> \frac{\lambda(a+1)}{b} \end{aligned} \tag{5.12}$$

c) $v_i - b - \sqrt{\Delta} > 0$

In this case, both r_{i1} and $r_{i2} > 0$.

The boundaries of v_i can be written as:

$$\begin{aligned} |v_i - b| &> 0 \\ (v_i - b)^2 &> \Delta \implies \\ b &< v_i < \frac{\lambda(a+1)}{b} \end{aligned} \tag{5.13}$$

Values of $J(\tau_i)$ at 0 and r_{i2} can be found as:

$$\begin{aligned} J(\tau_i = 0) &= \frac{1}{2\lambda} v_i^2 + (a+1)\log b \\ J(\tau_i = r_{i2}) &= \frac{1}{2\lambda}\left[\frac{-v_i - b_i + \sqrt{\Delta}}{2}\right]^2 + (a+1)\log\frac{v_i + b + \sqrt{\Delta}}{2}. \end{aligned} \tag{5.14}$$

This results in values for τ_i^* given by:

$$\tau_i^* = \begin{cases} 0 & \text{if } J(\tau_i = 0) < J(\tau_i = r_{i2}), \\ r_{i2} & \text{o.w.} \end{cases} \tag{5.15}$$

.

We will use the line search algorithm with the proximal gradient method to obtain τ^{k+1} and λ^k as shown in Table 5.4.3.

Following similar derivations from Chapter 4, we can find the derivative components as:

$$\frac{\partial f(\tau)}{\partial \tau_i} = \frac{1}{2}[S_i - Q_i^2] \tag{5.16}$$

and

$$\frac{\partial^2 f(\tau)}{\partial \tau_i^2} = \frac{1}{2}[-S_i^2 + 2Q_i^2 S_i], \tag{5.17}$$

where $Q_i = \sigma^{-2} x_i^T \tilde{y} - \sigma^{-4} x_i^T \tilde{X} \Sigma \tilde{X}^T \tilde{y}$, $S_i = \sigma^{-2} x_i^T x_i - \sigma^{-4} x_i^T \tilde{X} \Sigma \tilde{X}^T x_i$, $\Sigma = (A + \sigma^{-2} \tilde{X}^T \tilde{X})^{-1}$ and $A = diag(\tilde{\tau}_1^{-1} ... \tilde{\tau}_k^{-1})$.

The same expressions holds true for $f(\tau)$, μ, u and σ^2 as well.

$$f(\tau) = (\sigma^2)^{n-k} |\sigma^2 I_{k \times k} + \tilde{X}^T \tilde{X}^T| + \sigma^{-2} ||\tilde{y}||^2 - \sigma^{-4} \tilde{y}^T \tilde{X} \Sigma \tilde{X}^T \tilde{y}, \tag{5.18}$$

$$\mu^{k+1} = \frac{1^T C^{-1} y}{1^T C^{-1} 1}, \tag{5.19}$$

$$\sigma^{2(k+1)} = \frac{||\tilde{y} - \tilde{X} u||^2}{n - k + \sum_{i=1}^{k} \frac{[\Sigma]_{ii}}{\tilde{\tau}_i}}, \tag{5.20}$$

and

$$u = \sigma^{2(k)} \Sigma \tilde{X}^T \tilde{y}. \tag{5.21}$$

where 1 is a vector whose elements are all 1 and $C^{-1} = \sigma^{-2} I - \sigma^{-4} \tilde{X} \Sigma \tilde{X}^T$. Now that we have derived all the necessary expressions, we can summarize our EBlasso algorithm for the NEG model as shown in Table 5.4.3.

5.5 Experiment Procedure

As shown in [CHX11], the algorithm based on EBlasso coordinate ascent outperformed other state of the art algorithms such as EB [Xu07], RVM [Tip01, TF03] and

Table 5.1: Proximal method with line search algorithm

Begin
Let $\alpha = \lambda^{k-1}, \gamma \in (0,1)$
$\hat{f}_\alpha(z, v^k) \triangleq f(v^k) + \nabla f(v^k)^T(z - v^k) + \frac{1}{2\alpha}||z - v^k||_2^2$
while true **do**
$v^k = \left[\tilde{v}^k\right]^+ = \left[\tau^k - \lambda^k D^k \nabla f(\tau^k)\right]^+$
$z = prox_{\alpha g}(v^k)$
if $f(\tau) \leq \hat{f}_\alpha(z, v^k)$ **then**
break
end if
$\alpha = \gamma\alpha$
end while
$\lambda^k = \alpha$, $\tau^{k+1} = z$
End

Table 5.2: EBlaso with proximal gradient

Begin
Initialization: Choose $\lambda > 0$, $u = 1^T \frac{y}{n}$, $\tilde{y} = y - u1$, $\gamma = 0.5$, $\lambda^0 = 1$, $\sigma^2 = \frac{0.1||\tilde{y}||^2}{n}$
repeat
Begin
Subroutine: $(\tilde{\tau}^{k+1}, \Sigma, \tilde{X}) = proxm(\tau^k, \lambda^{k-1}, \gamma)$
End
Calculate μ^{k+1}
Calculate $\sigma^{2(k+1)}$
until Indices of elements of $\tilde{\tau}$ do not change AND $\frac{|\tilde{\tau}^k - \tilde{\tau}^{k-1}|^2}{|\tilde{\tau}^{k-1}|^2} < \epsilon$, where ϵ is a small number
End

LASSO [YX08, PC08] with both the power of detection as well as computational efficiency. Thus, in our simulation study we compared the performance of our new proximal gradient algorithm with the coordinate ascent algorithm. We implemented both the algorithms in C/C++ using the same level of coding and algorithmic techniques to have a fair comparison. All the simulations were run on a personal computer with Intel Xeon 3.39GHz CUP running 64 bit Windows server 2012 with 32GB of RAM.

We ran five phases of experiments. In the first phase, we simulated a single large chromosome of 2400 centimorgan (cM) long covered by evenly spaced 481 markers ($q = 481$) with a marker interval of 5 cM. The simulated population was an F_2 family derived from the cross of two inbred lines with a sample size 1000 ($n = 1000$). We only simulated the main effects in this dataset. However, both algorithms ran with full sets of parameters including epistatic effects, which results in a total of $1+481+\binom{481}{2} = 115,922$ parameters. We used the ten-fold cross validation technique to find out the optimal hyperparameter values for both algorithms.

In the second phase, to further analyze the robustness of our proximal gradient algorithm over various sample sizes, we used 100 simulated datasets with sample sizes 200, 400, 600, 800 and 1000. The optimal values of the hyperparameters for both algorithms were evaluated following the same ten-fold cross-validation technique described earlier.

So far we have considered only the main effects. To evaluate the performance of the new algorithms over an extremely higher number of model parameters, we created a dataset that included both main and epistatic effects. Basically, twenty markers

are QTLs with main effects and 20 out of the $\binom{481}{2} = 115,440$ marker pairs have interaction effects. This results in a QTL model containing a total of $1+481+\binom{481}{2} = 115,922$ possible effects. The PD, FDR and CUP time of the proximal method were compared with the coordinate ascent method. Based on the third phase results, we observed that the detection rate of our proximal gradient algorithm can be further improved by using a hybrid model. Therefore, in the fourth phase, we evaluated the performance of our hybrid model using the same dataset we used in phase three.

Finally, in phase five, we used a real dataset obtained from [LPD+07]. This dataset consisted of 150 samples of double haploids (DH) derived from the cross of two spring barley varieties, Morex and Steptoe. There were 495 markers distributed along seven pairs of chromosomes of the barley genome, covering 206 cM of the barley genome. This led to a total number of model effects of $1 + 495 + \binom{495}{2} = 122,761$, about 818 times as large as the sample size. We compared the detections from both algorithms based on different *p-values*.

5.6 Results and Discussion

5.6.1 Phase 1: Simulated Data

A total of $q = 481$ markers were simulated on a large single chromosome of 2400 centimorgan (cM) long with evenly spaced marker interval of 5 cM. The simulated population was an F_2 family derived from the cross of two inbred lines with a sample size $n = 1000$. The genotype indicator variable for individual i at marker j was

defined as $x_{ij} = 1, 0, -1$ for the three genotypes, A_1A_1, A_1A_2 and A_2A_2, respectively. Twenty markers were QTLs with 20 main effects. We did not simulate epistatic effects, environmental effects or gene-environment ($G \times E$) effects. However, both algorithms incorporated epistatic effects also in the model parameters, even though the dataset was only simulated for main effects. The true population mean was $\mu = 100$ and the residual variance was $\sigma_0^2 = 10$. Note that the QTLs were assumed to coincide with the markers. If QTLs are not on the markers, they may still be detected since correlation between a QTL and a nearby marker is high, although a slightly larger sample size may be needed to give the same power of detection.

The total phenotypic variance for the trait can be written as

$$\sigma_\gamma^2 = \sigma^2 + \sum_{j=1}^{q} \sum_{j'=1}^{q} \beta_j \beta_j' cov(x_j, x_j'), \tag{5.22}$$

where $cov(x_j, x_j')$ is the covariance between x_j and x_j' if $j \neq j'$ or the variance of x_j' if $j = j'$, which can be estimated from the data. If we ignore the contributions from the covariance terms, which are relatively small, the proposition of the phenotypic variance explained by a particular QTL effect j can be approximated by

$$h_j^2 = \frac{\beta_j^2 var(x_j)}{\sigma_\gamma^2}, \tag{5.23}$$

where $var(x_j)$ is the variance of X_j.

We used the prediction error (PE) [Tib96] obtained from ten-fold cross validation process to select the optimal values for hyperparameters a and b. For estimated phenotype value $\hat{y}_i = \hat{\mu} + \tilde{x}_i u$, the PE is calculated by:

$$PE = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{5.24}$$

Specifically, the dataset was first divided into ten subsets. Nine out of these ten subsets were used as the training data to estimate the model parameters, and the log likelihoods of the remaining testing data were calculated using the estimated parameters. This process was repeated ten times until every subset was tested. Since values of a and b are continuous, ideally we could have used a large number of discrete pair of values for these two variables in the cross-validation. However, that would have required a longer training time. Instead of using a large number of combinations, we used a similar technique used in [CHX11] to obtain a reduced number of candidate values for hyperparameters a and b.

Basically, we started with $a = b = 0.001, 0.01, 0.05, 0.1, 0.5, 1$. The degree of shrinkage generally decreases along this path. As shown in the Table 5.3, $a = .05$ and $b = .05$ had the lowest PE value for proximal gradient method. To check whether this is the optimal combination, we further ran cross-validation by fixing the b at .05, with $a = .1$ and $a = .01$. Usually, the degree of shrinkage decreases when a decreases. This effect is clearly shown in the Table 5.3.

The parameter combinations $(a = 0.001, b = 0.05)$ and $(a = 0.01, b = 0.05)$ were close, but noticeably higher than the $(a = 0.05, b = 0.05)$. Therefore, we selected the $a = 0.05$ and $b = 0.05$ as the optimal values from the cross-validation results. To further observe the variation of the true detections (TP) and false detections (FP) over various combinations of hyperparameter values used in ten-fold cross-validation, we also included TP and FP values in the Table 5.3. Note that the number of detections is based on the full dataset with all the 1000 samples, but not from cross-validation.

Since the comparison algorithm, the coordinate ascent, is also based on the NEG priors, it has the same type of hyperparameters. Therefore, we followed the same technique for that one as well. We also included the candidate hyperparameter value combinations that we used in ten-fold cross validation with the coordinate ascent algorithm in Table 5.3. Out of all the candidate parameter combinations, $a = 0.001$ and $b = 0.001$ had the lowest PE value for coordinate ascent algorithm. It was shown during the ten-fold cross validation that coordinate ascent took much longer time to complete the training than the proximal gradient method. This is because the proximal gradient method had a very fast convergence. The Table 5.4 shows the final detections based on optimal values for both algorithms.

As one can clearly see from Table 5.4, our algorithm outperformed the coordinate ascent algorithm in terms of detection rate. Basically, the proximal gradient algorithm detected three more effects than the coordinate ascent algorithm. Note that all the effects we detected from our new algorithm were based on $p\text{-}value \leq 0.05$. The non-detections shown with '*' were actually detected by our algorithm, but they were later filtered out by the *t-test* with $p\text{-}value = 0.05$. We will further analyze this phenomenon in Phase 4.

Clearly, the results show that the main contribution of our algorithm is to the performance aspect. Basically, our algorithm finished within 6.7 seconds, while the coordinate ascent algorithm took 38.4 seconds, meaning that our new algorithm is 5.7 times faster than the coordinate ascent algorithm. Specifically, in the proximal gradient method, the non-zero effects in the model are considered simultaneously, and global optimal is obtained faster and more accurate compared with the greedy

Table 5.3: Variaton of PE values over different combinations of a and b obtained from ten-fold cross validation

Algorithm	a	b	TP	FP	PE
	0.001	0.001	8	1	54.7815
	0.01	0.01	13	1	44.9232
	0.05	0.05	15	0	43.7188
	0.1	0.1	13	0	44.6603
Proximal	0.5	0.5	13	0	45.4422
Gradient	1.0	1.0	12	0	46.9343
	0.01	0.05	15	0	43.8779
	0.001	0.05	15	0	43.8733
	-0.01	0.05	15	0	44.0462
	0.001	0.001	12	0	80.1496
	0.01	0.01	14	0	83.915749
	0.05	0.05	14	0	83.929658
	0.1	0.1	14	0	83.922186
Coordinate	0.5	0.5	13	0	84.071424
Ascent	1.0	1.0	14	0	84.309103
	0.01	0.001	14	0	80.150408
	0.05	0.001	14	0	80.153684

Table 5.4: True estimated QTL effects for the simulated data with main effects.

Markers	Position (cM)	Proximal gradient (Is detected?)	Coordinate ascent (Is detected?)
11	50	Yes	Yes
26	125	No	Yes
42	205	No*	No
48	235	Yes	Yes
72	355	Yes	Yes
73	360	Yes	No
123	610	Yes	Yes
127	630	Yes	Yes
161	800	No*	No
181	900	Yes	No
182	905	Yes	Yes
185	920	Yes	No
221	1100	Yes	Yes
243	1210	Yes	Yes
262	1305	Yes	Yes
268	1335	No*	No
270	1345	Yes	No
274	1365	Yes	Yes
361	1800	Yes	Yes
461	2300	No*	No
CPU time		6.745 sec	38.417 sec

approach that considers one effect at a time used by the coordinate ascent algorithm. Since the effects are considered simultaneously, this allows most of the computational workflows to be implemented as parallel workflows, and that could further increase the performance. However, we did not implement any parallel execution workflows in our algorithm for the results presented here.

5.6.2 Phase 2: Effect Over Various Sample Sizes

In Phase 1, we observed that our proximal gradient method is capable of detecting QTLs more efficiently than the coordinate ascent with better detection accuracy. To further analyze the performance of our new algorithm over various sample sizes, we generated 100 replicates with sample sizes 200, 400, 600, 800 and 1000 using the same techniques we described earlier. Each replicate consists of individuals whose genotypes at 481 markers were independently generated and whose phenotypes were calculated from (5.2) with e independently generated from Gaussian random variables with zero mean and covariance 10. Only the main effects were considered. We used the same ten-fold cross validation technique we described earlier with each of these sampled datasets in obtaining the optimal hyperparameter values for both algorithms. The PD and FDR are plotted in Figures 5.1 and 5.2, respectively.

The figures are based on the average over 100 data samples. Both methods demonstrated 0.0 false discovery rate over all the sample sizes. As one can clearly see from Figure 5.1, the proximal gradient algorithm showed a better detection rate across all the sample sizes. Note that we used the $p\text{-}value = 0.05$ with all the simulated

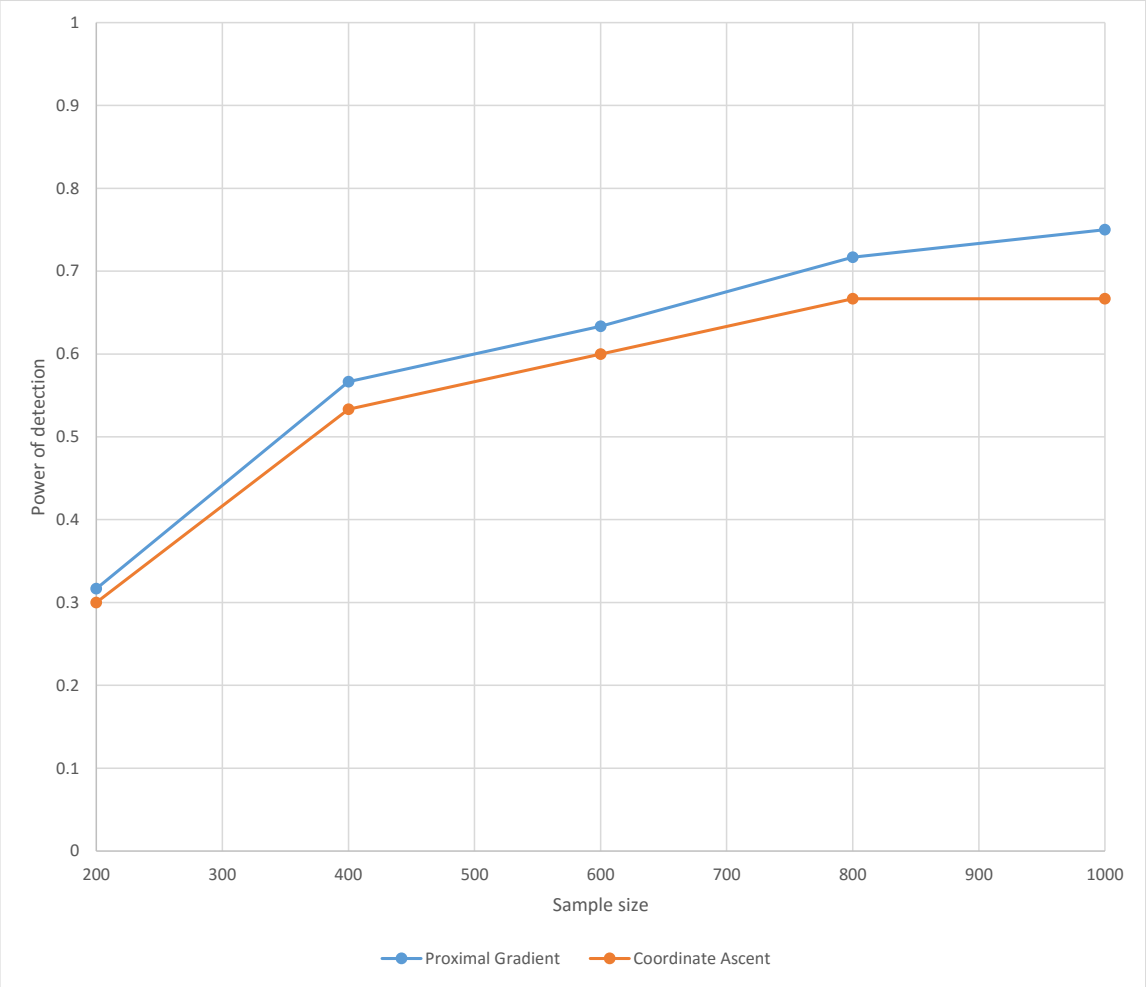

Figure 5.1: Power of detection for the proximal gradient and coordinate ascent algorithm. Performance data were obtained from the mean of 100 replicas for different sample sizes (n = 200, 400, 600, 800, 1000).

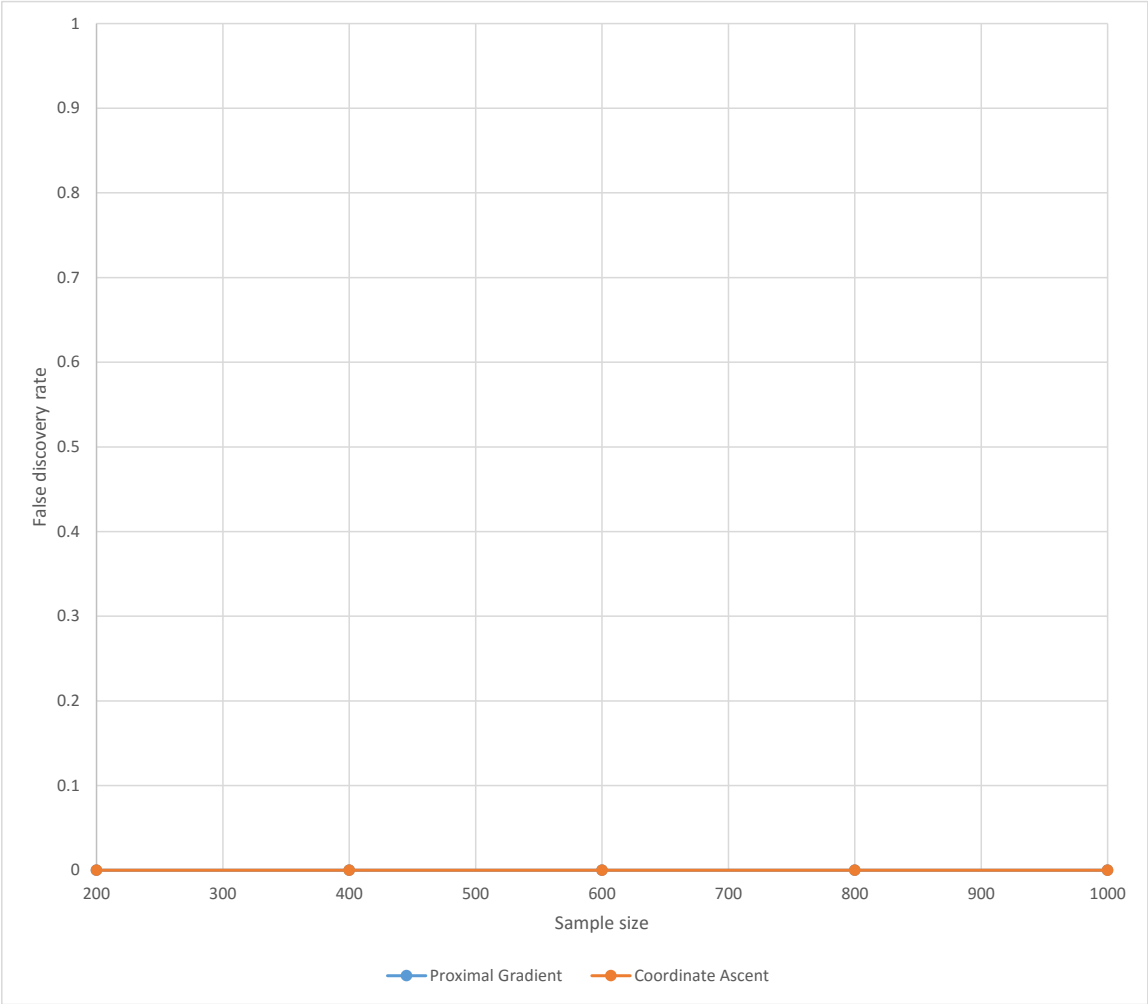

Figure 5.2: False discovery rate for the proximal gradient and coordinate ascent algorithm. Performance data were obtained from mean of 100 replicas for different sample sizes (n = 200, 400, 600, 800, 1000).

datasets. In general, when the sample size is larger, the power of detection increases, as indicated in Figure 5.1. As shown from the results, the main contribution of the proximal gradient method is to the performance improvement to the runtime. We have shown the average run time over various sample sizes in Table 5.5.

Table 5.5: Average run time over various sample sizes for proximal gradient method and the coordinate ascent method.

Sample size	Proximal Gradient CPU time (sec)	Coordinate Ascent CPU time (sec)	$\frac{t_{CoordinateAscent}}{t_{ProximalGradient}}$
200	1.109	5.076	4.575
400	2.208	16.755	7.588
600	3.499	24.191	6.914
800	3.673	33.024	8.992
1000	5.439	32.394	5.955

To further highlight the improvement in terms of runtime, we also plotted the performance characteristics of our algorithm in Figure 5.3. The *y-axis* represents the division of average run time of the coordinate ascent algorithm ($t_{CoordinateAscent}$) by the average run time of proximal gradient algorithm ($t_{ProximalGradient}$). As Figure 5.3 clearly indicates, our new algorithm is orders of magnitude faster than the coordinate ascent algorithm across all sample sizes. Furthermore, as we mentioned elsewhere, the performance of our new algorithm can be further improved by parallelizing calculation workflows. This is only possible with the new algorithm since it considers nonzero effects in the model simultaneously, rather than the greedy approach that considers one effect at a time.

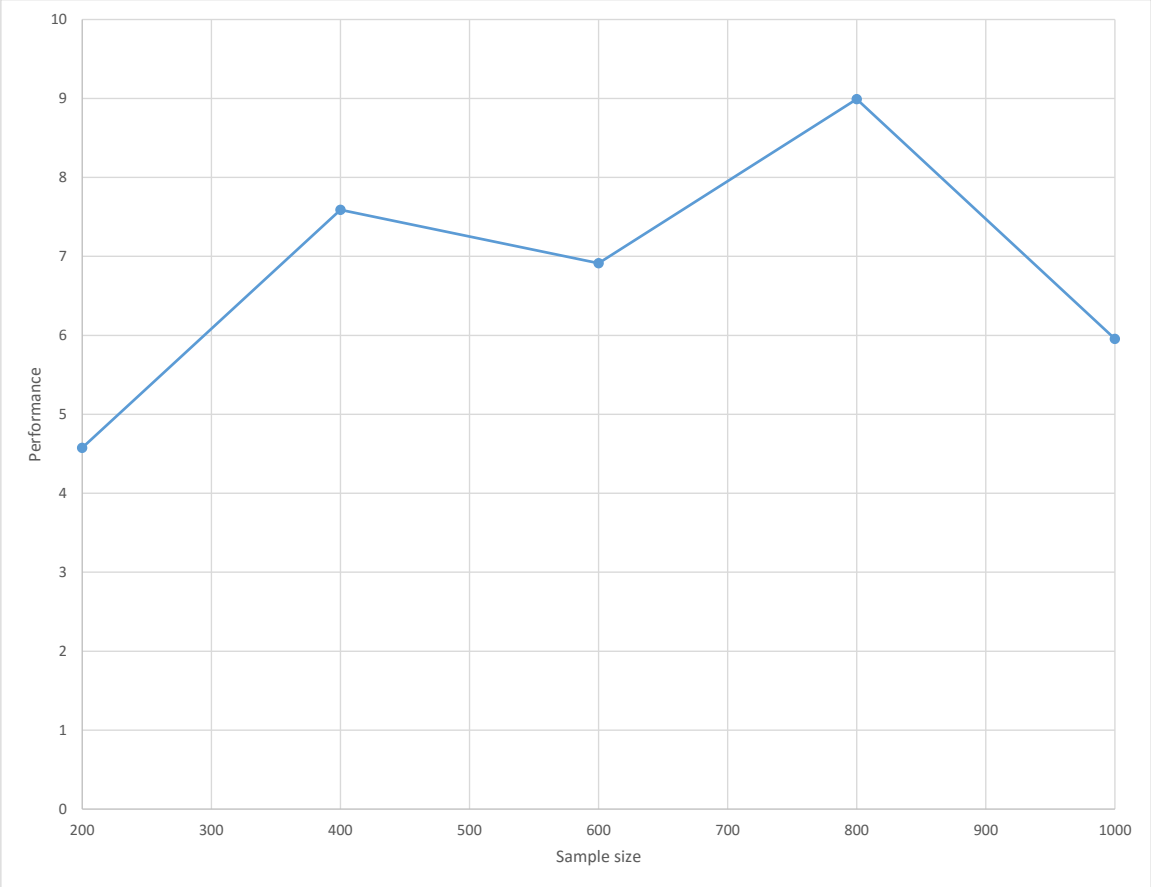

Figure 5.3: Performance is calculated as $t_{ProximalGradient}/t_{CoordinateAscent}$. Performance data were obtained by using the mean of 100 replicates.

5.6.3 Phase 3: Including Epistatic Effects

Usually with the real datasets, we find smaller sample sizes. To better capture this, we simulated a single large chromosome of 2400 cM long covered by evenly spaced q = 481 markers with a marker interval of 5 cM with interactive effects. The simulated population was an F2 family derived from the cross of two inbred lines with a sample size n=600. Twenty markers are QTLs with main effects and 20 out of the $\binom{481}{2} = 115,440$ marker pairs have interaction effects. Environmental effects and $G \times E$ effects were not simulated. The true population mean is $\mu = 100$ and the residual variance is $\sigma_0^2 = 10$. Including the population mean, the main and the pair-wise epistatic effects, the total number of model effects was $1 + 481 + \binom{481}{2} = 115,922$, about 193 times as large as the sample size.

We used ten-fold cross validation to estimate optimal values for hyperparameters. We used the same technique described in Phase 1 to reduce the number of candidate pairs for a and b. Table 5.6 shows the detection status for both algorithms based on the optimal hyperparameter values.

As shown in Table 5.6, the proximal gradient algorithm detected 14 true effects with only a single false effect, while the coordinate ascent algorithm detected 15 true effects with two false effects. Even though the new algorithms detected single effects less than the coordinate ascent algorithm, it detected 11 out of 14 effects from the exact location. However, with the coordinate ascent algorithm, it only detected 10 out of 15 from the exact location. The most noticeable improvement is in the CPU time. As one can see, the proximal gradient algorithm finished within 0.84 seconds

Table 5.6: True estimated QTL effects for the simulated data with main and epistatic effects for sample size 600.

Markers	Position (cM)	Proximal gradient (Is detected?)	Coordinate ascent (Is detected?)
(11,11)	(50,50)	Yes	Yes
(26,26)	(125,125)	Yes	Yes
(42,42)	(205,205)	No*	Yes
(48,48)	(235,235)	No*	No
(72,72)	(355,355)	Yes	No
(73,73)	(360,360)	Yes	Yes
(123,123)	(610,610)	No*	No
(127,127)	(630,630)	No	No
(161,161)	(800,800)	No	No
(181,181)	(900,900)	Yes	No
(182,182)	(905,905)	Yes	Yes
(185,185)	(920,920)	Yes	No
(221,221)	(1100,1100)	Yes	Yes
(243,243)	(1210,1210)	No*	No
(262,262)	(1305,1305)	No*	No
(268,268)	(1335,1335)	No	No
(270,270)	(1345,1345)	No	No
(274,274)	(1365,1365)	Yes	No
(361,361)	(1800,1800)	No	No
(461,461)	(2300,2300)	No	No
(5,6)	(20,25)	No	No
(6,39)	(25,190)	Yes	Yes
(42,220)	(205,1095)	Yes	Yes
(75,431)	(370,2150)	No	No
(81,200)	(400,995)	No*	No
(82,193)	(405,960)	No	No
(87,164)	(430,815)	No*	Yes
(87,322)	(430,1605)	Yes	Yes
(92,395)	(455,1970)	No*	Yes
(104,328)	(515,1635)	No	No
(118,278)	(585,1385)	No*	Yes
(150,269)	(745,1340)	No	No
(237,313)	(1180,1560)	No	No
(246,470)	(1225,2345)	No*	No
(323,464)	(1610,2315)	No	No
(328,404)	(1635,2015)	No*	Yes
(342,420)	(1705,2095)	No	No
(344,407)	(1715,2030)	No*	Yes
(373,400)	(1860,1995)	Yes	No
(431,439)	(2150,2190)	Yes	Yes
CPU time		0.84 sec	27.56 sec

while the coordinate ascent took 27.56 seconds. This means that the new proximal gradient algorithm is 32.8 times faster than the coordinate ascent algorithm. This is a very impressive performance improvement when dealing with much larger datasets with extremely large number of model parameters. All the detections from the new algorithm were based on the *p-value* of 0.05.

When we further analyzed the filtered out detections from the *t-test*, we observed that we had 27 true detections (marked with '*'). But 13 of them were filtered out by the *t-test*. We observed similar behavior in Phase 1 as well. This phenomenon gave us a direction to define a novel algorithm where we can further improve the detection rate, still keeping the runtime improvement at a high level. That is the hybrid model.

5.6.4 Phase 4: Hybrid Model

As we saw in the simulation results from Phase 3, our algorithm has extremely fast convergence with comparatively higher detection rate. However, some of the true detections were filtered out by the *t-test*. As an example, in the previous simulations, there were 27 detections, but 13 of them were filtered out by the *t-test*. To still keep the majority of detections without sacrificing the false discovery rate, we propose a hybrid model where we utilize the main strengths of our algorithm while alleviating the problem of filtering done by the *t-test* using a secondary algorithm. Basically, our proximal algorithm does the heavy lifting where it reduces the parameter space to a very small manageable set within a couple of seconds. As an example, in the previous simulations, the original number of parameters was 115,922 and our algorithm was

capable of reducing it to 122 nonzero parameters within 0.84 seconds with 27 true detections. Out of those 122, there were few false positives, but the majority of them were collocated with the true detections.

Since the parameter space for the secondary algorithm is bound to the output from the main algorithm, there is no scalability concern for the secondary algorithm. To this end, we can redefine the model to the secondary algorithm as:

$$y = \mu + X'\beta' + e, \tag{5.25}$$

where X' is the design matrix associated with the output from our main algorithm and β' is the effect of markers. Note that, X' is a matrix with size $n \times k'$ where $k' \ll p + \frac{q}{2}(q+1) + pq$, which is the original parameter space. To solve this problem, we can properly choose some prior distribution and define the joint posterior distribution of the parameters. Then we can utilize any technique to solve this as another optimization problem such as MCMC. As we saw in the chapter 2, the MCMC methods incur higher computational cost.

We borrowed the greedy algorithm used in [CHX11] as our secondary algorithm and modified and optimized it to work with a smaller sets of data. One of the motivations was that we could reuse some of the implementation we had already done for our main algorithm. Since scalability is not a concern here, we can do most of the calculations in memory. We chose proper data structures to facilitate this and implemented with C/C++ that yields higher efficiency. With this hybrid model, we can redefine our algorithm as shown in Table 5.6.4.

To evaluate our hypothesis, we used the same dataset we used in Phase 3. Since the hybrid model consists of two stages, we first did the ten-fold cross validation with the proximal gradient method to obtain the optimal values for hyperparameters. Then we fixed those parameters and initiated another ten-fold cross validation process for the secondary algorithm. Since the secondary algorithm runs with an extremely small number of parameters, the variation of the results was expected to be small, meaning that it is less sensitive to parameter values. This is a key aspect since it will drastically reduce the number of candidate values for cross-validations. To demonstrate this clearly, we tabled the PE values for various parameter combinations, $a = b = 0.01$, 0.05, 0.1, 0.5 and 0.8 in Table 5.8. We also included TP and FP detections to back up our theory. Note that this data is shown only the output from the secondary algorithm, but not the final merged results.

As one can see from Table 5.8, the output from the secondary algorithm detected 19 true effects with only a single false effect at (257,257). Even that one is just 25 cM away from the QTL located at (262, 262). As shown in Table 5.8, various values of secondary algorithm result in the same level of detection. This is what we claimed before, that we could have used any secondary algorithm for this purpose. However, with this secondary algorithm, the original detections were greatly discounted as well. As one can see, the number of TP drops from 27 to 19, because the secondary algorithm we used had strong shrinkage. Since the number of parameters that the secondary algorithm needs to handle was largely reduced, we could have used a different algorithm with lower shrinkage to increase the number of TPs while keeping the FPs low.

Table 5.7: EBlaso With Proximal Gradient Hybrid

Begin

Initialization: Choose $\lambda > 0$, $u = 1^T \frac{y}{n}$, $\tilde{y} = y - u1$, $\gamma = 0.5$, $\lambda^0 = 1$, $\sigma^2 = \frac{0.1||\tilde{y}||^2}{n}$

repeat

Begin

Subroutine: $(\tilde{\tau}^{k+1}, \Sigma, \tilde{X}) = proxm(\tau^k, \lambda^{k-1}, \gamma)$

End

Calculate μ^{k+1}

Calculate $\sigma^{2(k+1)}$

until Indices of elements of $\tilde{\tau}$ do not change AND $\frac{|\tilde{\tau}^k - \tilde{\tau}^{k-1}|^2}{|\tilde{\tau}^{k-1}|^2} < \epsilon$, where ϵ is a small number

Obtain the final indices of elements of $\tilde{\tau}$ and the corresponding design matrix X'

Begin

Subroutine: $(\tau_{final}) = secondary(\tilde{\tau}, X', y)$

End

Merge results

End

Table 5.8: Various PE values for the secondary algorithm with hybrid model.

$a = b$	TP	FP	PE	CPU time (sec)
0.01	18	1	1.864	0.91
0.05	19	1	1.070	0.92
0.1	19	1	0.928	0.81
0.5	19	1	1.014	0.97
0.8	19	1	0.974	0.95

The final CPU time for the hybrid model is the summation of the original proximal gradient method and the secondary algorithm, which was 1.65 seconds. This means that the new hybrid model is still 16.7 times faster than the coordinate ascent algorithm. In this way, we slightly sacrificed performance in favor of better detection rate. Table 5.9 shows the final detections from the hybrid model. For ease of comparison, we also include the results from the vanilla proximal gradient method as well.

As one can see from Table 5.9, the final hybrid model detected 22 true effects. When compared with the coordinate ascent algorithm, our hybrid model was capable of detecting seven more true effects. This gives PD and FDR for the proximal hybrid method, 0.55 and 0.04, respectively. When compared to the coordinate ascent algorithm, it had PD and FDR values, 0.375 and 0.12, respectively. Furthermore, the new hybrid model is 16 times faster than the coordinate ascent algorithm. As mentioned earlier, we could improve performance further by parallelizing calculation workflows. At the same time, we could increase the detection rate by deriving a new secondary algorithm with slightly less shrinkage. Since the requirement for the secondary algorithm is fairly simple, there is more room for improvements. Basically, we can use some closed form solutions that may lead to better accuracy since performance is not a concern for the secondary algorithm due to greatly reduced parameter space.

Table 5.9: True estimated QTL effects for the simulated data with main and epistatic effects for hybrid model.

Markers	Position (cM)	Proximal gradient	Proximal-gradient hybrid	Coordinate ascent
(11,11)	(50,50)	Yes	Yes	Yes
(26,26)	(125,125)	Yes	Yes	Yes
(42,42)	(205,205)	No	Yes	No
(48,48)	(235,235)	No	Yes	No
(72,72)	(355,355)	Yes	Yes	No
(73,73)	(360,360)	Yes	Yes	Yes
(123,123)	(610,610)	No	No	No
(127,127)	(630,630)	No	No	No
(161,161)	(800,800)	No	No	No
(181,181)	(900,900)	Yes	Yes	No
(182,182)	(905,905)	Yes	Yes	Yes
(185,185)	(920,920)	Yes	Yes	No
(221,221)	(1100,1100)	Yes	Yes	Yes
(243,243)	(1210,1210)	No	No	No
(262,262)	(1305,1305)	No	No	No
(268,268)	(1335,1335)	No	No	No
(270,270)	(1345,1345)	No	No	No
(274,274)	(1365,1365)	Yes	Yes	No
(361,361)	(1800,1800)	No	No	No
(461,461)	(2300,2300)	No	No	No
(5,6)	(20,25)	No	No	No
(6,39)	(25,190)	Yes*	Yes*	Yes
(42,220)	(205,1095)	Yes	Yes	Yes*
(75,431)	(370,2150)	No	No	No
(81,200)	(400,995)	No	Yes*	No
(82,193)	(405,960)	No	No	No
(87,164)	(430,815)	No	Yes	Yes*
(87,322)	(430,1605)	Yes	Yes	Yes*
(92,395)	(455,1970)	No	Yes*	Yes*
(104,328)	(515,1635)	No	No	No
(118,278)	(585,1385)	No	Yes*	Yes
(150,269)	(745,1340)	No	No	No
(237,313)	(1180,1560)	No	No	No
(246,470)	(1225,2345)	No	No	No
(323,464)	(1610,2315)	No	No	No
(328,404)	(1635,2015)	No	Yes*	Yes
(342,420)	(1705,2095)	No	No	No
(344,407)	(1715,2030)	No	Yes*	Yes*
(373,400)	(1860,1995)	Yes*	Yes*	No
(431,439)	(2150,2190)	Yes*	Yes*	Yes
CPU time		0.84 sec	1.65 sec	27.56 sec

5.6.5 Phase 5: Real Data

So far, we have analyzed the performance of our proximal algorithm based on simulated datasets. To see how well the algorithm behaves with the real data, we used a dataset obtained from Luo et al. [LPD$^+$07]. This dataset consists of $n = 150$ double haploids (DH) derived from the cross of two spring barley varieties, Morex and Steptoe. The total number of markers was $q = 495$ distributed along seven pairs of chromosomes of the barley genome, covering 206 cM of the barley genome. The phenotype was the spot blotch resistance measured as the lesion size on the leaves of barley seedlings. Note that spot blotch is a fungus named *Cochliobolus sativus*. The genotype of the markers was encoded as +1 for genotype A (the Morex parent), -1 for genotype B (the Steptoe parent), and 0 for missing genotype.

Ideally, the missing genotypes should be imputed from known genotypes of neighboring markers. For simplicity, we replaced the missing genotypes with 0 in order to use the phenotypes of the individuals with missing genotypes. The total missing genotypes only account for about 4.2% of all the genotypes. Including the population mean, the main and the pair-wise epistatic effects, the total number of model effects was $1 + 495 + \binom{495}{2} = 122,761$, about 818 times as large as the sample size.

We used five-fold cross validation since the sample size was too small to do ten-fold cross validation. For the hyperparameters, we chose values from the set $0.001, .01, .05, .07, .08, .1, .5, .8, .9, 1$ where $a = b$. The table 5.10 shows the *PE* values for each combination. We used two *p-values*, 0.05 and 0.01 in our simulations. Basi-

cally, *p-value* of 0.01 gives more conservative results. The table 5.10 shows detections for both of these *p-values*.

Table 5.10: Number of detections and PE values for different hyperparameter combinations for real data.

$a = b$	PE	Effects *p-value*= 0.05	Effects *p-value*= 0.01	CPU time seconds
0.001	2.2192	59	21	0.81
0.01	2.3359	42	15	0.80
0.05	2.4408	29	10	0.80
0.07	2.4438	28	8	0.90
0.08	2.4438	28	7	0.65
0.1	2.4885	27	7	0.59
0.5	2.5232	21	6	0.48
0.8	2.5282	20	5	0.42
0.9	2.5282	20	5	0.43
1.0	2.5282	20	5	0.57

Note that the number of detections was obtained from the full sets of samples. The PE values shown are for the *p-value* of 0.05. We also included CPU time for each parameter combination. As shown on the Table 5.10, our proximal algorithm took less than a second across all combinations. Furthermore, the variation of PE was very small over a range of hyperparameter combinations. Therefore, we calculated the frequency of every effect across all parameter combinations. The effects that have more than or equal to 50% occurrences are shown in Table 5.11 for *p-value* = 0.05. Similar information for results with *p-value* = 0.01 is shown in Table 5.12.

According to Tables 5.11 and 5.12 , our proximal algorithm detected twenty effects common to all parameter combinations for *p-value*= 0.05 and five effects for *p-value*= 0.01. For the coordinate ascent algorithm, we used parameter combinations $a = b =$

Table 5.11: Number of detections across all hyperparameter combinations for *p-value*= 0.05

Marker	Frequency
(1,467)	10
(30,417)	10
(31,335)	10
(50,72)	10
(58,288)	10
(78,149)	10
(79,147)	10
(109,141)	10
(110,395)	10
(113,274)	10
(137,334)	10
(145,379)	10
(154,274)	10
(202,384)	10
(223,296)	10
(259,292)	10
(330,359)	10
(426,442)	10
(427,432)	10
(435,439)	10
(400,477)	7
(9,253)	6
(48,165)	6
(106,284)	6
(302,329)	6
(305,435)	6
(137,331)	5
(291,438)	5

$0.001, 0.01, 0.05, 0.07, 0.1, 0.5, 1.0$. Table 5.13 summarizes the results based on the five-fold cross validations.

As one can see from Table 5.11, the proximal gradient method was more consistent over a large range of parameter values. On the other hand, the coordinate ascent algorithm seems to either under-detect or over-detect on most of the parameter combinations. Since coordinate ascent had only one or two detections for $a = b = 0.001, 0.01$, we did a similar frequency calculation for $a = b = 0.05, 0.07, 0.1, 0.5$ and 1.0. Eight effects were common to these parameter combinations.

To demonstrate runtime improvement of the proximal algorithm, we plotted runtime over various parameter combinations in Figure 5.4. Clearly, in Figure 5.4, the new proximal gradient method is considerably faster than the coordinate ascent algorithm across all combinations. As an example, based on the combination that resulted in the lowest PE value, the proximal gradient method is 115 times faster than the coordinate ascent algorithm.

Table 5.12: Number of detections across all hyperparameter combinations for p-$value$= 0.01

Marker	Frequency
(78,149)	10
(113,274)	10
(223,296)	10
(426,442)	10
(435,439)	10
(154,274)	7
(58,288)	6

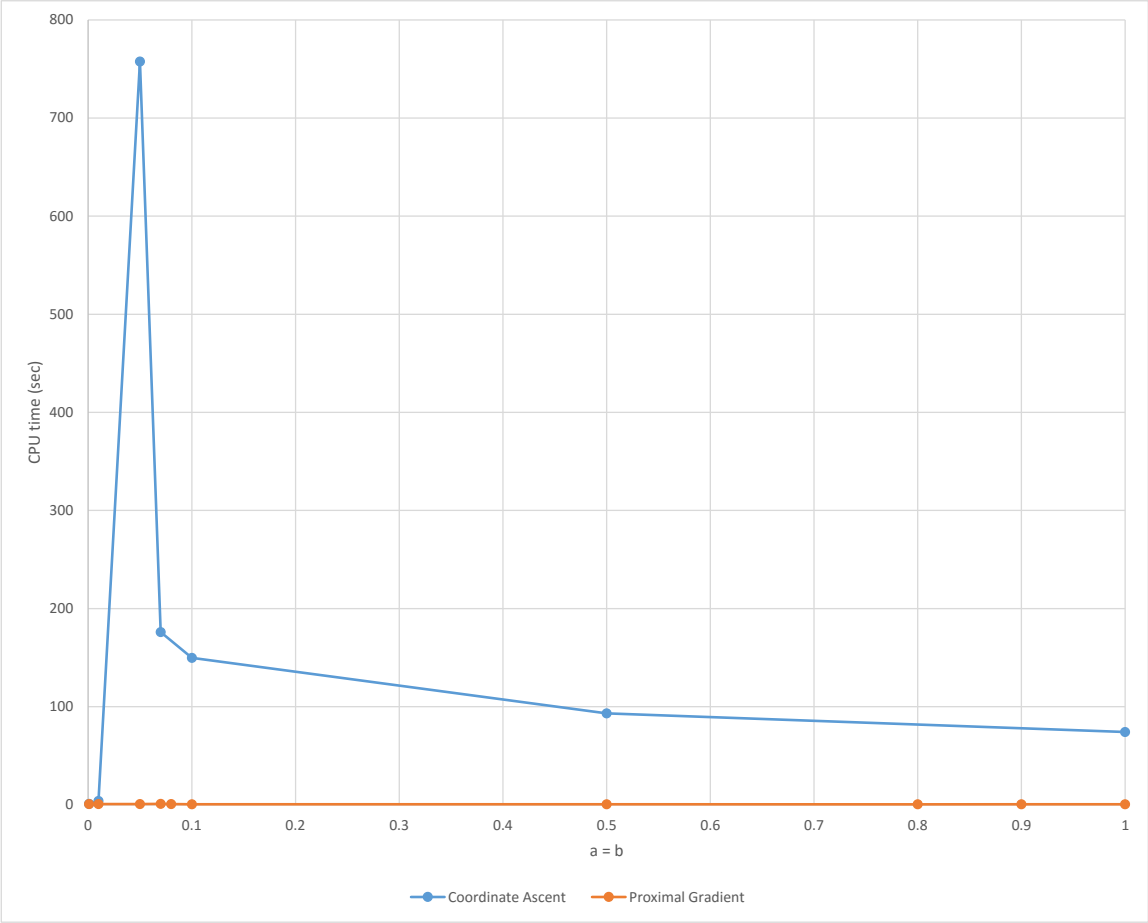

Figure 5.4: CPU time for real data over various hyperparameter value combinations.

Table 5.13: Number of detections and PE values for different hyperparameter combinations for coordinate ascent

$a = b$	PE	Effects $p\text{-}value = 0.05$	Effects $p\text{-}value = 0.01$	CPU time seconds
0.001	1.979627	1	1	0.89
0.01	2.006083	2	2	3.85
0.05	2.610536	11	11	757.56
0.07	2.621296	15	10	175.94
0.1	2.513092	128	13	149.81
0.5	2.501755	112	9	93.27
1.0	2.483921	115	8	74.25

5.7 Summary

There are various methods used in multiple QTL mapping such as EB [Xu07], RVM [Tip01, TF03], Lasso [YX08, PC08], penalized likelihood (PENaL) [ZX05] and stochastics search variable selection(SVVS) [YGA03]. According to the comparison provided by Xu [Xu07], the SSVS method is much slower than the EB method; whereas, Lasso and PENAL methods are faster than the EB method. Although EB, Lasso, PENAL and SSVS methods all produced satisfactory results in a simulation study [Xu07], the EB method outperformed the other three methods. Moreover, when applied to a real data set, the EB and Lasso detected some effects, whereas PENAL and SSVS failed to generate any meaningful results [Xu07].

On the other hand, based on the simulation study by Cai et al. [CHX11], the EBlasso method detected more true effects than the EB method without raising the false positive rate. Furthermore, when analyzing a real dataset, they found that the EBlasso method detected a reasonable number of effects, but the EB method

detected one or zero effect depending on the values of the hyperparameters used. These observations in both simulation study and real data analysis demonstrate that the EBlasso method outperforms both the EB and the Lasso methods.

The EBlasso method was built upon the idea of the RVM in machine learning. The EBlasso and EB methods, as well as the RVM, all are based on a Bayesian hierarchical linear regression model and all estimate the variances of the regression coefficients. The difference of the three methods in the regression model is the different prior distributions for the hyperparameters. The EB method and the RVM employ inverse χ^2 and uniform distributions, respectively, for the variances of the regression coefficients, while the EBlasso assigns exponential distributions to the variance components and uses a Gamma distribution for the parameter of the exponential distribution. The uniform prior distribution used by the RVM may not provide enough degree of shrinkage for certain data, and thus generate a large number of false positive effects, as shown in [Xu10, CHX11].

The prior distributions used by the EBlasso and RVM methods enable one to estimate the variance components in a closed form, while the EB method generally needs a numerical optimization algorithm to estimate the variance components. For this reason, EBlasso and RVM methods can always find the unique optimal estimate of a variance component with much less computation than the EB method [CHX11]. Since the EBlasso method claims to have superior detection power with extremely fast performance over EB, RVM and LASSO methods, and EB method outperforms most of the other state-of-the-art methods, we compared our new proximal gradient algorithm with the EBlasso method.

The greedy coordinate ascent algorithm used in the EBlasso method considers one effect at a time in an iterative fashion [CHX11]. This is the main drawback of this method, where it could still take longer time even though it is much faster than the EB method. Moreover, it fails to capture much broader complex relations across multiple model parameters simultaneously.

Capitalizing on this idea, in this chapter we have developed a novel proximal gradient optimization algorithm for the EBlasso method. Our model is based on a Bayesian hierarchical linear regression model that uses three-level hierarchical prior for the regression coefficients, namely, normal, exponential and Gamma (NEG). It first estimates the variance of the regression coefficients and then finds the posterior distribution of the regression coefficients based on the estimated variance. Compared to the previous implementation with the greedy coordinate ascent algorithm that adds, removes or reestimates one variable from the model at a time, the proximal gradient algorithm considers all variables simultaneously and maps points toward the minimum of the objective function in (5.5). In the proximal algorithm, the base operation is to evaluate the proximal operator, which is typically a specialized method with closed form solutions [PB13].

The simulation results demonstrate that the new proximal gradient algorithm provides superior performance in terms of power of detection and false positive rate. It is also clearly shown that the proximal gradient algorithm is orders of magnitude faster than the coordinate ascent algorithm based on all the simulated variants. Our new algorithm can easily handle more than 10^5 possible effects within a few seconds running on an average personal computer.

Our new algorithm has shown impressive performance over real data as well. As one can see from the results in Phase 5, the new algorithm reliably detected more effects than the coordinate ascent algorithm with all the hyperparameter value combinations. Based on the optimal hyperparameter values, the new algorithm is 115 times faster than the coordinate ascent algorithm.

With the novel proximal gradient hybrid algorithm, we detected seven more true effects than the coordinate ascent algorithm, which yields PD and FDR values of 0.55 and 0.04, respectively, over 0.375 and 0.12 provided by the coordinate ascent algorithm. Moreover, the proximal gradient algorithm was 16 times faster than the coordinate ascent algorithm based on the optimal parameter values. We emphasize here that the impressive detection rates were obtained with small number of samples.

Since our proximal gradient algorithm considers nonzero effects in the model parameters simultaneously, we can further improve the speed by parallelizing calculation workflows. Furthermore, we can derive new algorithms to be used as the secondary algorithm with the hybrid model. Since the secondary algorithm runs on very small sets of model parameters, we can easily find better closed form solutions that could yield increased detection rates.

Our recent studies demonstrated that EBlasso has a broad range of applications, such as whole-genome QTL mapping and pathway-based genome-wide association study (GWAS) [HXC14b, HMVC14]. When the number of possible effects is very large in QTL models, with both main and epistatic effects, the computation times become a critical concern. Given that high density marker maps can be easily obtained, thanks

to advancement in sequencing technology, it would be worthwhile to explore these areas with the new methods developed in this chapter.

CHAPTER 6

Conclusion and Future Work

In Chapter 1, we explored two main research areas: gene regulation and genotype/phenotype association. Along those two major avenues, we formulated two main problems: TFBSs identification and QTL mapping. We further observed various methods used in these two areas, including state-of-the-art techniques. In Chapter 2, we developed a novel MRF based TFBSs identification algorithm and further improved the performance by introducing a pairwise-MRF model in Chapter 3. We diverged from the TFBSs identification problem and developed a novel proximal based algorithm for EBlasso-NE model to infer QTLs in Chapter 4. In Chapter 5, we further extended our algorithm to EBlasso-NEG model, which demonstrated extremely fast convergence with higher detection rate. Furthermore, we developed a novel proximal gradient hybrid model to further enhance the detection rate with lower false positive rate. We next summarize the contribution of the proposed methods, and discuss future research avenues in this concluding chapter.

6.1 Conclusion

Given the fundamental significance of gene regulation and genotype/phenotype association in understanding the genetic basis of complex biological systems, powerful statistical modeling and inference methods are highly desirable. In this dissertation, we propose novel MRF based methods for the TFBSs identification problem and further improve the performance by developing a pairwise-MRF model. The exhaustive simulation study affirmed that our new methods are fast and accurate when compared to the other state-of-the-art methods.

Toward the QTL mapping problem, we propose novel empirical Bayesian methods that are able to model a large number of additive and dominance markers effects, including both main and epistatic effects. Furthermore, the novel proximal gradient based algorithms are capable of handling model parameters simultaneously over the greedy approach used by other state-of-the-art methods. This also opens up further improvements to the performance by allowing the algorithm implementers to execute most of the workflows in parallel by taking full advantage over advanced computer technologies.

Both simulation and real data analysis suggested that the proposed EBlasso-NE [AHC14], EBlasso-NEG and EBlasso-NEG hybrid methods are fast and accurate variable selection and estimation methods for multiple QTL mapping. The performance of the proposed EBlasso methods was compared to the other state-of-the-art genotype and phenotype association methods, such as EB [Xu07], RVM [Tip01, TF03], Lasso [YX08, PC08], penalized likelihood (PENaL) [ZX05] and stochastics

search variable selection(SVVS) [YGA03] as well as our previous EBlasso coordinate ascent method [CHX11].

In addition to the fast convergence and improved detection rate, the newly proposed EBlasso algorithms estimate both posterior mode and covariance of the marker effects, which in turn enables statistical testing. We demonstrated this fact by utilizing the *t-statistics* with EBlasso in QTL mapping. The EBlasso algorithms apply prior assumption to the regression coefficients and are applicable to different regression models.

In summary, given the fundamental importance of gene expression and genotype/phenotype associations in understanding the genetic basis of complex biological systems, the MRF, pairwise-MRF, EBlasso-NE, EBlasso-NEG and EBlasso-NEG hybrid algorithms and software packages developed in this dissertation achieve the effectiveness, robustness and efficiency needed for successful application to biology. With the advancement of high-throughput molecular technologies in generating information at genetic, epigenetic, transcriptional and posttranscriptional levels, the methods developed in this dissertation can have broad applications to infer TFBSs and different types of genotype and phenotype associations.

6.2 Future Work

Capitalizing on the sparse models and the associated inference methods presented in this dissertation, we envision the following new research directions.

6.2.1 MRF-based Discriminative Methods for Discovering DNA Motifs

As described in Chapter 3, our method for DNA motif discovery first learns an MRF model from a set of known motif sequences, and then the likelihood of each candidate sequence belonging to the motif is evaluated based on the MRF model and compared with the likelihood a background sequence. An immediate work is to extend our method to *de novo* motif discovery. In this case, we have a set of DNA sequences and need to find a DNA motif from this set of sequences without any information about the motif. Many algorithms have been developed for *de novo* motif discovery. A well-known algorithm is MEME [BWML06], where the expectation-maximization (EM) method was exploited to estimate the position weight matrix (PWM) of the motif. If we use an MRF model instead of a PWM for the motif, we can use our MRF-based approach for *de novo* discovery of a motif.

Early *de novo* motif discovery methods such as MEME find over-represented sequences in a set of sequences that contain motifs, relative to background sequence model. A more recent discriminative approach, in contrast, searches for specific motifs that are present at a higher frequency in a positive set of sequences than in a negative set of sequences. The discriminative approach can improve the accuracy of prediction motifs by reducing the false positive rate and increasing detection power. A frequency difference in the positive and negative sets is required for accurate classification of one set from the other, and is often not reflected by over-representation in the positive set, as many instances of the over-represented motifs

are also found in the negative set. This can reduce the false positive rate. On the other hand, motif sequences may be much less likely to appear in the negative set than that predicted by a background model. Apparently, the discriminative approach increases the power of detection. A number of discriminative methods have been developed [Bai11, HBS+10, GPGK13, HZS+11, YMF+14, PS14, FBS08, RB07]; however, they do not take into account the interaction between nucleotides in a motif, which may limit the accuracy of predicting motifs. To overcome this problem, we can use MRF to model a motif and develop an MRF-based discriminative method, which is expected to improve motif prediction accuracy.

6.2.2 Empirical Bayesian Lasso for QTL Mapping of Binary Traits

In Chapter 5, we developed a very efficient empirical Bayesian (EB) Lasso algorithm based on the proximal operator for QTL mapping of quantitative traits. Although complex binary traits only show binary phenotypic variation, different from the continuous variation in quantitative traits, they do not follow a simple Mendelian pattern of inheritance and also have a polygenic basis similar to that of quantitative traits. Therefore, like QTL mapping for quantitative traits, mapping for complex binary traits aims to identify multiple genomic loci that are associated with the trait and to estimate the genetic effects of these loci, possibly including any of the following effects: main effects, gene-gene interactions (epistatic effects) and effects of gene-environment interactions. In [HXC13], we employed a Bayesian logistic regression

model as the QTL model for binary traits that includes both main and epistatic effects. Our logistic regression model employs hierarchical priors for regression coefficients similar to the ones used in the Bayesian LASSO linear model for multiple QTL mapping for continuous traits described in Chapter 5. We developed efficient empirical Bayesian algorithms to infer the logistic regression model. We expect that our proximal operator-based method developed in Chapter 5 can also be extended to handle binary traits.

6.2.3 Empirical Bayesian Method for Inference of Gene Networks

Genes in living organisms do not function in isolation, but may interact with each other forming complicated networks [LRR+02]. Uncovering the structure of gene networks is critical to understanding gene functions and cellular dynamics, as well as to system-level modeling of individual genes and cellular functions. A number of computational methods have been developed to infer gene networks from gene expression data based on the pair-wise correlation between the expression levels of each pair of genes [BTS+00, BMS+05], Gaussian graphical models [DHJ+04, SS04], Bayesian networks [FLNP00, SSR+03], linear regression models [GdBLC03, dBTG+05, SS05].

More recently, genetic variations and gene expression data are exploited jointly for inference of gene networks based on Bayesian networks [ZLL+04, ZWZ+07, ZZS+08], likelihood test [KJ06, CESS07, NFAY08, AFLH08, MZZS09], and the structural equation model (SEM) [XLF04, LdlFH08, LM10, MEW+10, CBG]. As naturally occur-

ring genetic variations provide perturbations to gene networks, exploiting such genetic variation apparently will improve the accuracy of inferring network structures. More important, such genetic variations enable inference of the causal relationship between different genes or between genes and certain phenotypes.

Motivated by the fact that gene networks or more general biochemical networks are sparse [TYHC03, JMBO01, THPRCV98], we employed a sparse SEM to infer gene networks from both gene expression and eQTL data. Incorporating network sparsity constraints, a sparsity-aware maximum likelihood (SML) algorithm is developed for network topology inference. Computer simulations demonstrate that the SML algorithm offers significantly better performance than existing competing algorithms. We expect that a Bayesian SEM can be used to model sparse gene networks, and that our proximal operator-based approach can be adopted to infer such a Bayesian SEM. This approach will not only provide an efficient algorithm for inferring gene networks, but also enable incorporation of prior knowledge into the inference process, which may further improve the inference accuracy.

Bibliography

[AAS00] T. Akutsu, H. Arimura, and S. Shimozono, *On approximation algorithms for local multiple alignment*, RECOMB 2000, ACM, 2000, pp. 1–7.

[ABH+03] B. Alberts, D. Bray, K. Hopkin, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Essential cell biology*, second ed., Garland Pub, 2003.

[AC10] K. L. Ayers and H. J. Cordell, *SNP selection in genome-wide and candidate gene studies via penalized logistic regression*, Genet. Epidemiol **34** (2010), 879–891.

[AFLH08] J. E. Aten, T. F. Fuller, A. J. Lusis, and S. Horvath, *Using genetic markers to orient the edges in quantitative trait networks: The NEO software*, BMC Syst Biol **2** (2008), 34.

[AGM+90] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, *Basic local alignment search tool*, J Mol Biol **215** (1990), 403–410.

[AHC14] I. P. K. Appuhamilage, A. Huang, and X. Cai, *Fast proximal gradient optimization of the empirical Bayesian Lasso for multiple quantitative trait locus mapping*, IEEE Conf. on Signal and Information Processing (GlobalSIP), December 2014, pp. 1348–1351.

[AJL+14] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, *Molecular biology of the cell*, sixth ed., Garland Science, 2014.

[AJX01] J. Akey, L. Jin, and M. Xiong, *Haplotypes vs single marker linkage disequilibrium tests: what do we gain*, Eur. J. Hum **9** (2001), no. 4, 291–300.

[AKFST09] S. S. Murray A. K. Frazer, N. J. Schork, and E. J. Topol, *Human genetic variation and its contribution to complex traits*, Nature Reviews Genetics **10** (2009), 241–251.

[Arm55] P. Armitage, *Tests for linear trends in proportions and frequencies*, Biometrics **11** (1955), 375–386.

[Bai11] T. L. Bailey, *DREME: Motif discovery in transcription factor ChIP-seq data*, Bioinformatics **27** (2011), 16531659.

[Bal06] D. J. Balding, *A tutorial on statistical methods for population association studies*, Nat. Rev. Genet **7** (2006), 781–791.

[BBC08] D. J. Balding, M. Bishop, and C. Cannings, *Handbook of statistical genetics*, third ed., Wiley, 2008.

[BE94] T. L. Bailey and C. Elkan, *Fitting a mixture model by expectation maximization to discover motifs in biopolymers*, Proc Int Conf Intell Syst Mol Biol., 1994, pp. 28–36.

[BE95] ———, *Unsupervised learning of multiple motifs in biopolymers using expectation maximization*, Machine Learning **21** (1995), 51–80.

[BEFK03] Y. Barash, G. Elidan, N. Friedman, and T. Kaplan, *Modeling dependencies in protein-DNA binding sites*, In Proc. of RECOMB-03, 2003, pp. 28–37.

[BGMS03] I. Ben-Gal, G. Morag, and A. Shmilovici, *Context-based statistical process control: a monitoring procedure for state-dependent processes*, Technometrics **45** (2003), 293–311.

[BGSG+05] I. Ben-Gal, A. Shani, A. Gohr, J. Grau, S. Arviv, A. Shmilovici, S. Posch, and I. Grosse, *Identification of transcription factor binding sites with variable-order Bayesian networks*, Bioinformatics **21** (2005), no. 11, 2657–2666.

[BH95] Y. Benjamini and Y. Hochberg, *Controlling the false discovery fate: a practical and powerful approach to multiple testing*, J. R. Stat. Soc. **57** (1995), 289–300.

[Bis06] C. M. Bishop, *Graphical models for machine learning and digital communication*, Springer, 2006.

[BJC02] M. L. Bulyk, P. L. F. Johnson, and G. M. Church, *Nucleotides of transcription factor binding sites exert interdependent effects on the binding affinities of transcription factors*, Nucleic Acids Research **30** (2002), no. 5, 1255–1261.

[BLFS01] P. V. Benos, A. S. Lapedes, D. S. Fields, and G. D. Stormo, *Samie: statistical algorithm for modeling interaction energies*, PSB'01 (2001), 115–126.

[BMS+05] K. Basso, A. A. Margolin, G. Stolovitzky, U. Klein, R. Dalla-Favera, and A. Califano, *Reverse engineering of regulatory networks in human B cells*, Nature Genetics **37** (2005), 382–390.

[BN03] T. L. Bailey and W. S. Noble, *Searching for statistically significant regulatory modules*, Bioinformatics (Proceedings of the European Conference on Computational Biology) **19** (2003), ii16–ii25.

[Bow03] P. J. Bowler, *Evolution: the history of an idea*, Berkeley: University of California Press, 2003.

[BPQ+06] M. F. Berger, A. A. Philippakis, A. M. Qureshi, F. S. He, P. W. Estep, and M. L. Bulyk, *Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities*, Nature Biotechnology **24** (2006), 1429–1435.

[BT09] A. Beck and M. Teboulle, *A fast iterative shrinkage-thresholding algorithm for linear inverse problems*, SIAM Journal on Imaging Sciences **2** (2009), no. 1, 183–202.

[BTS+00] A. J. Butte, P. Tamayo, D. Slonim, T. R. Golub, and I. S. Kohane, *Discovering functional relationships between RNA expression and chemotherapeutic susceptibility using relevance networks*, Natl Acad Sci **97** (2000), no. 22, 1218212186.

[Bul03] M. L. Bulyk, *Computational prediction of transcription-factor binding site locations*, Genome Biology **5** (2003), no. 1, 201–211.

[BW91] W. L. Buntine and A. S. Weigend, *Bayesian back-propagation*, Complex Syst. **5** (1991), 603–643.

[BW99] P. Bhlmann and A. J. Wyner, *Variable length markov chains*, Ann. Statist **27** (1999), no. 2, 480–513.

[BWML06] T. L. Bailey, N. Williams, C. Misleh, and W. W. Li, *MEME: discovering and analyzing DNA and protein sequence motifs*, Nucleic acids research **34** (2006), no. supp1 2, 369–373–9.

[CBG] X. Cai, J. A. Bazerque, and G. B. Giannakis, *Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations*, PLoS Comput Biol **23**, no. 9, e1003068.

[CDKK00] D. Cai, A. Delcher, B. Kao, and S. Kasif, *Modeling splice sites with bayes networks*, Bioinformatics **16** (2000), no. 2, 152–158.

[CESS07] L. S. Chen, F. Emmert-Streib, and J. D. Storey, *Harnessing naturally randomized transcription to infer regulatory relationships among genes*, Genome Biol **8** (2007), no. 10, R219.

[CG04] R. Castelo and R. Guig, *Splice site identification by idlBNs*, Bioinformatics **20** (2004), i69–i76.

[CH04] O. Carlborg and C. S. Haley, *Epistasis: too often neglected in complex trait studies*, Nature Reviews Genetics **5** (2004), 618–625.

[CHX11] X. Cai, A. Huang, and S. Xu, *Fast empirical Bayesian LASSO for multiple quantitative trait locus mapping*, BMC Bioinformatics **12** (2011), no. 1, 211.

[Coc54] C. C. Cockerham, *An extension of the concept of partitioning hereditary variance for analysis of covariances among relatives when epistasis is present*, Genetics **39** (1954), 859–882.

[Con07] T. I. H. Consortium, *A second generation human haplotype map of over 3.1 million snps*, Nature **449** (2007), 851–861.

[Cor02] H. J. Cordell, *Epistasis: what it means, what it doesn't mean, and statistical methods to detect it in humans*, Hum. Mol. Genet **11** (2002), 2463–2468.

[Cor09] ———, *Detecting gene-gene interactions that underlie human diseases*, Nat. Rev. Genet. **10** (2009), 392–404.

[Cri74] F. Crick, *The double helix: a personal view*, Nature **248** (1974), no. 5451, 799–9.

[CT06] G. C. Cawley and N. L. C. Talbot, *Gene selection in cancer classification using sparse logistic regression with Bayesian regularization*, Bioinformatics **22** (2006), no. 19, 2348–2355.

[Dal04] R. N. Dale, *A simple correction for multiple testing for sngle-nucleotide polymorphisms in linkage disequilibrium with each other*, Am. J. Hum. Genet **74** (2004), 765–769.

[dBTG+05] D. di Bernardo, M. J. Thompson, T. S. Gardner, S. E. Chobot, and E. L. Eastwood, *Chemogenomic profiling on a genome-wide scale using reverse-engineered gene networks*, Nat Biotechnol **23** (2005), 377383.

[DHJ+04] A. Dobra, C. Hans, B. Jones, J. R. Nevins, G. Yao, and M. West, *Sparse graphical models for exploring gene expression data*, Multivar Analysis **90** (2004), no. 1, 196212.

[DPS04] M. Dudk, S. J. Phillips, and R. E. Schapire, *Performance guarantees for regularized maximum entropy density estimation*, Proceedings of the 17th Annual Conference on Computational Learning Theory, Springer Verlag, 2004, pp. 472–486.

[DRW03] B. Devlin, K. Roeder, and L. Wasserman, *Analysis of multilocus models of association*, Genet Epidemiol **25** (2003), 36–47.

[DSS03] M. Djordjevic, A. M. Sengupta, and B. I. Shraiman, *A biophysical approach to transcription factor binding site discovery*, Genome Res. **13** (2003), no. 11, 2381–2390.

[EG01] W. J. Ewens and G. R. Grant, *Statistical methods in bioinformatics: An introduction*, Springer-Verlag, New York, INc., 2001.

[EP02] E. Eskin and P. A. Pevzner, *Finding composite regulatory patterns in DNA sequences*, Bioinformatics **18** (2002), no. Suppl 1, S354–S63.

[FBS08] F. Fauteux, M. Blanchette, and M. V. Strmvik, *Seeder: discriminative seeding DNA motif discovery*, Bioinformatics **24** (2008), no. 20, 2303–2307.

[FF93] I. E. Frank and J. H. Friedman, *Statistical view of some chemometrics regression tools*, Technometrics **35** (1993), 109–135.

[FG53] R. E. Franklin and R. G. Gosling, *Molecular configuration in sodium thymonucleate*, Nature **171** (1953), no. 4356, 740–1.

[FH97] J. W. Fickett and A. G. Hatzigeorgiou, *Eukaryotic promoter recognition*, Genome Res. **7** (1997), no. 9, 861–878.

[FLNP00] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, *Using Bayesian network to analyze expression data*, Comput Biol **7** (2000), no. 3-4, 601620.

[FM96] D. S. Falconer and T. F. C. Mackay, *ntroduction to quantitative genetics*, third ed., Addison-Wesley, 1996.

[Fu98] W. J. J. Fu, *Penalized regressions: The bridge versus the Lasso*, J.Comput. Graph. Stat **7** (1998), 397–416.

[GB11] J. E. Griffin and P. J. Brown, *Bayesian hyper-lassos with non-convex penalization*, Aust. N. Z. J. Stat **53** (2011), 423–442.

[GCS+13] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian data analysis*, third ed., Chapman and Hall/CRC, 2013.

[GdBLC03] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins, *Inferring genetic networks and identifying compound mode of action via expression profiling*, Science **301** (2003), 102105.

[Gel06] A. Gelman, *Prior distributions for variance parameters in hierarchical models*, Bayesian Anal **1** (2006), 515–533.

[GES85] D. J. Galas, M. Eggert, and M. S.Waterman, *Rigorous pattern-recognition methods for DNA sequences: Analysis of promoter sequences from e. coli*, J. Mol. Biol **186** (1985), no. 1, 117–128.

[GH06] N. M. Gericke and M. Hagberg, *Definition of historical models of gene function and their relation to students understanding of genetics*, Science and Education **16** (2006), 849–881.

[GJPS08] A. Gelman, A. Jakulin, M. G. Pittau, and Y. Su, *A comparative investigation of methods for logistic regression with separated or nearly separated data*, Ann. Appl. Stat **2** (2008), 1360–1383.

[GPGK13] J. Grau, S. Posch, I. Grosse, and J. Keilwagen, *A general approach for discriminative de novo motif discovery from high-throughput data*, Nucleic Acids **41** (2013), no. 21, e197.

[GR81] M. M. Garner and A. Revzin, *A gel electrophoresis method for quantifying the binding of proteins to specific DNA regions: application to components of the escherichia coli lactose operon regulatory system*, Nucleic Acids Res **9** (1981), no. 13, 3047–3060.

[Gus97] D. Gusfield, *Algorithms on strings, trees and sequences: Computer science and computational biology*, Cambridge University Press, 1997.

[HBS+10] S. Heinz, C. Benner, N. Spann, E. Bertolino, Y. C. Lin, P. Laslo, J. X. Cheng, C. Murre, H. Singh, and C. K. Glass, *Simple combinations of lineage-determining transcription factors prime cis-regulatory elements required for macrophage and B cell identities*, Mol Cell **38** (2010), no. 4, 576–589.

[HCMF08] T. Hesterberg, N. H. Choi, L. Meier, and C. Fraley, *Least angle and l1 penalized regression: A review*, Stat. Surv **2** (2008), 61–93.

[Hei06] G. Heinze, *A comparative investigation of methods for logistic regression with separated or nearly separated data*, Statist. Med **25** (2006), 4216–4226.

[HETC00] J. D. Hughes, P. W. Estep, S. Tavazoie, and G. M. Church, *Computational identification of cis-regulatory elements associated with groups of functionally related genes in saccharomyces cerevisiae*, J. Mol. Biol **296** (2000), no. 5, 1205–1214.

[HETZ07] H. Huang, C. D. Eversley, D. W. Threadgill, and F. Zou, *Quantitative trait loci mapping for complex traits using markers of the entire genome*, Genetics **176** (2007), no. 4, 2529–2540.

[HGC95] D. Heckerman, D. Geiger, and D. M. Chickering, *Learning Bayesian networks: The combination of knowledge and statistical data*, Machine Learning **20** (1995), 197–243.

[HHG83] R. Harr, M. Haggstrom, and P. Gustafsson, *Search algorithm for pattern match analysis of nucleic acid sequences*, Nucleic Acids Res. **11** (1983), no. 9, 2943–2957.

[HK70] A. E. Hoerl and R. W. Kennard, *Ridge regression: Biased estimation for nonorthogonal problems*, Technometrics **12** (1970), 55–67.

[HL13] A. Huang and D. Liu, *From quantitative trait locus mapping to genomic selection: the roadmap towards a systematic genetics*, OA Genetics **1** (2013), no. 4, 1–5.

[HMVC14] A. Huang, E. Martin, J. Vance, and X. Cai, *Detecting genetic interactions in pathway-based genome-wide association studies*, Genet Epidemiol **38** (2014), no. 4, 300–309.

[HS06] F. Hoti and M. J. Sillanp, *Bayesian mapping of genotype x expression interactions in quantitative and qualitative traits*, Heredity **97** (2006), 4–18.

[HTF09] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction*, second ed., New York, Springer, 2009.

[HTRM10] A. Huang, M. Teplitski, B. Rathinasabapathi, and L. Ma, *Characterization of arsenic resistant bacterial communities in the rhizosphere of an arsenic hyperaccumulator pteris vittata*, Can. J. Microbiol **56** (2010), 236–246.

[HWIB08] C. J. Hoggart, J. C. Whittaker, M. Iorio, and D. J. Balding, *Simultaneous analysis of all snps in genome-wide and re-sequencing association studies*, PLoS Genet **4** (2008), e1000130.

[HXC13] A. Huang, S. Xu, and X. Cai, *Empirical Bayesian LASSO-logistic regression for multiple binary trait locus mapping*, BMC Genet **14** (2013), no. 1, 5.

[HXC14a] ———, *Empirical Bayesian elastic net for multiple quantitative trait locus mapping*, Heredity (2014), 79.

[HXC14b] ———, *Whole-genome quantitative trait locus mapping reveals major role of epistasis on yield of rice*, PLoS ONE **9** (2014), no. 1, e87330.

[HZS+11] P. Huggins, S. Zhong, I. Shiff, R. Beckerman, O. Laptenko, C. Prives, M. H. Schulz, I. Simon, and Z. Bar-Joseph, *A general approach for discriminative de novo motif discovery from high-throughput data*, Bioinformatics **27** (2011), no. 17, 2361–2367.

[IUP86] IUPAC, *Nomenclature committee of the international union of biochemistry (nc-iub). nomenclature for incompletely specified basis in nucleic acid sequences. recommendations 1984*, Natl. Acad. Sci. USA 83, 1986, pp. 4–8.

[JCIL05] E. R Jolly, C. S. Chin, I.Herskowitz, and H. Licorresponding, *Genome-wide identification of the regulatory targets of a transcription factor using biochemical characterization and computational genomic analysis*, BMC Bioinformatics **6** (2005), no. 275, 275–287.

[JLZL04] S. T. Jensen, X. S. Liu, Q. Zhou, and J. S. Liu, *Computationl discovery of gene regulatory binding motifs: A Bayesian perspective*, Statistical Science **19** (2004), no. 1, 188–204.

[JMBO01] H. Jeong, S. P. Mason, A. L. Barabassi, and Z. N. Oltvai, *Lethality and centrality in protein networks*, Nature **411** (2001), 4142.

[Kea98] M. J. Kearsey, *The principles of QTL analysis (a minimal mathematics approach)*, Experimental Botany **49** (1998), 1619–1623.

[KFL01] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, *Factor graphs and the sum-product algorithm*, IEEE Transactions on Information Theory **47** (2001), no. 2, 498–519.

[KGR+03] A. E. Kel, E. Gssling, I. Reuter, E. Cheremushkin, O. V. Kel-Margoulis, and E. Wingender, *Match: A tool for searching transcription factor binding sites in DNA sequences*, Nucleic Acids Res. **31** (2003), no. 13, 3576–3579.

[KJ06] D. C. Kulp and M. Jagalur, *Causal inference of regulator-target pairs by gene mapping of expression phenotypes*, BMC Genet **7** (2006), 125.

[KS80] R. Kindermann and J. L. Snell, *Markov random fields and their applications*, AMS, 1980.

[Kul59] S. Kullback, *Information theory and statistics*, A Wiley publication in mathematical statistics, 1959.

[LB89] E. S. Lander and D. Botstein, *Mapping mendelian factors underlying quantitative traits using rflp linkage maps*, Genetics **121** (1989), 185–199.

[LBL01] X. Liu, D. L. Brutlag, and J. S. Liu, *Bioprospector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes*, Pac Symp Biocomput **6** (2001), 127–138.

[LdlFH08] B. Liu, A. de la Fuente, and I. Hoeschele, *Gene network inference via structural equation modeling in genetical genomics experiments*, Genetics **178** (2008), 17631776.

[LGA+09] E. S. Lander, P. Green, J. Abrahamson, A. Barlow, M. J. Daly, S. E. Lincoln, and L. A. Newberg, *Mapmaker: an interactive computer package for constructing primary genetic linkage maps of experimental and natural populations*, Genomics **1** (2009), no. 2, 174–181.

[LGK07] S. Lee, V. Ganapathi, and D. Koller, *Efficient structure learning of markov networks using l1-regularization*, Advances in Neural Information Processing Systems **18** (2007), 817–824.

[LLF+09] S. Li, Q. Lu, W. Fu, R. Romero, and Y. Cui, *A regularized regression approach for dissecting genetic conflicts that increase disease risk in pregnancy*, Stat Appl Genet Mol Biol **8** (2009), no. 1, 1–28.

[LM10] B. A. Logsdon and J. Mezey, *Gene expression network reconstruction by convex feature selection when incorporating genetic perturbations*, PLoS Comput Biol **6** (2010), no. 12, e1001014.

[LNL95] J. S. Liu, A. F. Neuwald, and C. E. Lawrence, *Bayesian models for multiple local sequence alignment and gibbs sampling strategies*, Journal of the American Statistical Association **90** (1995), no. 432, 1156–1170.

[LPD+07] Z. W. Luo, E. Potokina, A. Druka, R. Wise, R. Waugh, and M. J. Kearsey, *SEP genotyping from affymetrix arrays is robust but largely detects cisacting expression regulators*, Genetics **176** (2007), 789–800.

[LR90] C. E. Lawrence and A. A. Reilly, *An expectation maximization (em) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences*, Proteins **7** (1990), 41–51.

[LRR+02] T. I. Lee, N. J. Rinaldi, F. Robert, D. T. Odom, and Z. Bar-Joseph, *Transcriptional regulatory networks in saccharomyces cerevisiae*, Science **298** (2002), 799804.

[LW98] M. Lynch and B. Walsh, *Genetics and analysis of quantitative traits*, first ed., Sunderland, MA, Sinauer, 1998.

[LX12] S. Lee and E. P. Xing, *Leveraging input and output structures for joint mapping of epistatic and marginal eQTLs*, Bioinformatics **28** (2012), no. 12, i137–i146.

[Men65] J.G. Mendel, *Versuche ber plflanzenhybriden verhandlungen des naturforschenden vereines in brnn*, Bd. IV fr das Jahr (1865), 3–47.

[MEW+10] X. Mi, K. Eskridge, D. Wang, P. S. Baenziger, B. T. Campbell, K. S. Gill, I. Dweikat, and J. Bovaird, *Regression-based multi-trait QTL mapping using a structural equation model*, Stat Appl Genet Mol Biol **9** (2010), no. 1, 1544.

[MFG+03] V. Matys, E. Fricke, R. Geffers, E. Gossling, M. Haubrock, R. Hehl, K. Hornischer, D. Karas, A. E. Kel, O. V. Kel-Margoulis, D. U. Kloos, S. Land, B. Lewicki-Potapov, H. Michael, R. Munch, I. Reuter, S. Rotert, H. Saxel, M. Scheer, S. Thiele, , and E. Wingender, *Transfac: Transcriptional regulation, from patterns to profiles*, Nucl. Acids Res. **31** (2003), 374–378.

[MS00] L. Marsan and M. F. Sagot, *Algorithms for extracting structured motifs using a suffix tree with an application to promoter and regulatory site consensus identification*, J Comput Biol. **7** (2000), 345–362.

[MS01] T. K. Man and G. D. Stormo, *Non-independence of mnt repressor-operator interaction determined by a new quantitative multiple fluorescence relative affinity (qumfra) assay*, Nucleic Acids Res. **29** (2001), no. 12, 2471–2478.

[MW05] J. H. Moore and S. M. Williams, *Traversing the conceptual divide between biological and statistical epistasis: systems biology and a more modern synthesis*, BioEssays **27** (2005), 637–646.

[MZZS09] J. Millstein, B. Zhang, J. Zhu, and E. E. Schadt, *Disentangling molecular relationships with a causal inference test*, BMC Genet **10** (2009), 23.

[NFAY08] E. C. Neto, C. T. Ferrara, A. D. Attie, and B. S. Yandell, *Inferring causal phenotype networks from segregating populations*, Genetics **179** (2008), 10891100.

[Ng04] A.Y. Ng, *Feature selection, l1 vs. l2 regularization, and rotational invariance.*, In Proc. 21st International Conference on Machine Learning, 2004.

[NM65] J. A. Nelder and R. Mead, *A simplex method for function minimization*, Comput J **7** (1965), 308–313.

[OFPK02] Y. L. Orlov, V. P. Filippov, V. N. Potapov, and N. A. Kolchanov, *Construction of stochastic context trees for genetic texts*, In Silico Biol. **2** (2002), no. 3, 233–247.

[OH99] U. Ohler and S. Harbeck, *Interpolated markov chains for eukaryotic promoter recognition*, 1999.

[ON01] U. Ohler and H. Niemann, *Identification and analysis of eukaryotic promoters: recent computational approaches*, Trends Genet. **58** (2001), 56–60.

[OPLS05] R. A. O'Flanagan, G. Paillard, R. Lavery, and A. M. Sengupta, *Non-additivity in protein-DNA binding*, Bioinformatics **21** (2005), no. 10, 2254–2263.

[OS09] R. B. O'Hara and M. J. Sillanpaa, *A review of bayesain variable selection methods: what, how and which*, Bayesian Anal **4** (2009), 85–118.

[PB13] N. Parikh and S. Boyd, *Proximal algorithms. foundations and trends in optimization*, no. 3, 123–231.

[PC08] T. Park and G. Casella, *The Bayesian Lasso*, J Am Stat Assoc **103** (2008), no. 482, 681–686.

[Pea88] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, California, 1988.

[Pea06] H. Pearson, *Genetics: what is a gene*, Nature **441** (2006), no. 7092, 398401.

[Pen07] E. Pennisi, *Genomics. DNA study forces rethink of what it means to be a gene*, Science **316** (2007), no. 5831, 15561557.

[PL88] W. R. Pearson and D. J. Lipman, *Improved tools for biological sequence comparison*, Proc Natl Acad Sci U S A. **85** (1988), no. 8, 2444–2448.

[PMMP04] G. Pavesi, P. Mereghetti, G. Mauri, and G. Pesole, *Weeder web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes*, Nucleic Acids Res. **1** (2004), no. 32, W199–W203.

[PMP04] G. Pavesi, G. Mauri, and G. Pesole, *In silico representation and discovery of transcription factor binding sites*, Briefings in Bioinformatics **5** (2004), no. 3, 217–236.

[PPL97] S. D. Pietra, V. D. Pietra, and J. Lafferty, *Inducing features of random fields*, IEEE Transactions on Pattern Analysis and Machine Intelligence **19** (1997), no. 4, 380–393.

[Pra08] L. A. Pray, *Discovery of DNA structure and function: Watson and crick*, Nature Education **1** (2008), no. 1, 1–8.

[PS14] R. Y. Patel and G. D. Stormo, *Discriminative motif optimization based on perceptron training*, Bioinformatics **30** (2014), no. 7, 941–948.

[RB07] E. Redhead and T. L. Bailey, *Discriminative motif discovery in DNA and protein sequences using the deme algorithm*, BMC Bioinformatics **8** (2007), 385.

[RC04] C. R. Robert and G. Casella, *Monte carlo statistical methods*, New York, Springer, 2004.

[RFJ+98] E. Roulet, I. Fisch, T. Junier, P. Bucher, and N. Mermod, *Evaluation of computer tools for the prediction of transcription factor binding sites on genomic DNA*, Silico Biol **1** (1998), no. 1, 21–28.

[Ris83] J. Rissanen, *A universal data compression system*, IEEE Transactions on Information Theory **29** (1983), 656– 664.

[Rit11] M. D. Ritchie, *Using biological knowledge to uncover the mystery in the search for epistasis in genome-wide association studies*, Ann. Hum. Genet **75** (2011), 172–182.

[RRW+00] B. Ren, F. Robert, J. J. Wyrick, O. Aparicio, E. G. Jennings, I. Simon, J. Zeitlinger, J. Schreiber, N. Hannett N, and E. Kanin, *Genome-wide location and function of DNA binding proteins*, Bioinformatics (Proceedings of the European Conference on Computational Biology) **290** (2000), 2306–2309.

[Sal97] S. L. Salzberg, *A method for identifying splice sites and translational start sites in eukaryotic mrna*, Comput Appl Biosci. **13** (1997), no. 4, 365–376.

[SBG07] A. Shmilovici and I. Ben-Gal, *Using a vom model for reconstructing potential coding regions in est sequences*, Computational Statistics **22** (2007), no. 1, 49–69.

[Sch78] G. Schwarz, *Estimating the dimension of a model*, Ann. Stat **6** (1978), 461–464.

[SH89] G. D. Stormo and G. W. Hartzell, *Identifying protein-binding sites from unaligned DNA fragments*, Proc Natl Acad Sci U S A. **86** (1989), no. 4, 1183–1187.

[SK03] S. K. Shevade and S. S. Keerthi, *A simple and efficient algorithm for gene selection using sparse logistic regression*, Bioinformatics **19** (2003), no. 17, 2246–2253.

[Sla14] J. M. W. Slack, *Genes-A very short introduction*, Oxford University Press, 2014.

[SS04] J. Schfer and K. Strimmer, *An empirical bayes approach to inferring large-scale gene association networks*, Bioinformatics **21** (2004), no. 6, 754764.

[SS05] ———, *A shrinkage approach to large-scale covariance matrix estimation and implications for functional genomics*, Stat Appl Genet Mol Biol **4** (2005), no. article 32.

[SSGE82] G. D. Stormo, T. D. Schneider, L. Gold, and A. Ehrenfeucht, *Use of the perceptron algorithm to distinguish translational initiation sites in e. coli*, Nucleic Acids Res. **10** (1982), no. 9, 2997–3011.

[SSR+03] E. Segal, M. Shapira, A. Regev, D. Pe'er, and D. Botstein, *Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data*, Nat Genet **34** (2003), 166–176.

[ST02] S. Sinha and M. Tompa, *Discovery of novel transcription factor binding sites by statistical overrepresentation*, Nucleic Acids Res. **15** (2002), no. 30, 5549–5560.

[Ste12] K. V. Steen, *Travelling the world of gene-gene interactions*, Brief. Bioinform **13** (2012), no. 1, 1–19.

[Sto00] G. D. Stormo, *DNA binding sites: representation and discovery*, Bioinformatics **16** (2000), no. 1, 16–23.

[TF03] M. E. Tipping and A. C. Faul, *Fast marginal likelihood maximisation for sparse Bayesian models*, Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, 2003, pp. 3–6.

[TG90] C. Tuerk and L. Gold, *Systematic evolution of ligands by exponential enrichment: Rna ligands to bacteriophage t4 DNA polymerase*, Nucleic Acids Res **249** (1990), no. 4968, 505–510.

[THPRCV98] D. Thieffry, A. M. Huerta, E. Perez-Rueda, and J. Collado-Vides, *From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in escherichia coli*, Bioessays **20** (1998), 433440.

[Tib94] R. Tibshirani, *Regression shrinkage and selection via the Lasso*, Journal of the Royal Statistical Society, Series B **58** (1994), 267–288.

[Tib96] ———, *Regression shrinkage and selection via the Lasso*, J. Roy. Stat. Soc. B. Met **58** (1996), 267–288.

[Tip01] M. E. Tipping, *Sparse Bayesian learning and the relevance vector machine*, J Mach Learn Res **1** (2001), 211–244.

[TLB+05] M. Tompa, N. Li, T. L. Bailey, G. M. Church, B. De Moor, E. Eskin, A. V. Favorov, M. C. Frith, Y. Fu, W. J. Kent, V. J. Makeev, A. A. Mironov, W. S. Noble, G. Pavesi, G. Pesole, M. Rgnier, N. Simonis, S. Sinha, G. Thijs, J. van Helden, M. Vandenbogaert, Z. Weng, C. Workman, C. Ye, and Z. Zhu Z, *Assessing computational tools for the discovery of transcription factor binding sites*, Nat Biotechnol **23** (2005), no. 1, 137–144.

[TLM+01] G. Thijs, M. Lescot, K. Marchal, S. Rombauts, B. De Moor, P. Rouz, and Y. Moreau, *A higher-order background model improves the detection of promoter regulatory elements by gibbs sampling*, Bioinformatics **17** (2001), no. 12, 1113–1122.

[TYHC03] J. Tegner, M. K. Yeung, J. Hasty, and J. J. Collins, *Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling*, Proc Natl Acad Sci **100** (2003), no. 10, 59445949.

[UMFK02] I. A. Udalova, R. Mott, D. Field, and D. Kwiatkowski, *Quantitative prediction of nf-kappa b DNA-protein interactions*, Proc Natl Acad Sci U S A **99** (2002), no. 12, 8167–8172.

[VBW98] L. Vandenberghe, S. Boyd, and S. P. Wu, *Determinant maximization with linear matrix inequality constraints*, SIAM Journal on Matrix Analysis and Applications **19** (1998), no. 2, 499–533.

[WAG84] M. S. Waterman, R. Arratia, and D. J. Galas, *Pattern recognition in several sequences: consensus and alignment*, Bull Math Biol. **46** (1984), no. 4, 515–527.

[Wat04] J. D Watson, *Molecular biology of the gene*, Cold Spring Harbour Laboratory Press, 2004.

[WCH+09] T. T. Wu, Y. F. Chen, T. Hastie, E. Sobel, and K. Lange, *Genome-wide association analysis by Lasso penalized logistic regression*, Bayesian Anal **25** (2009), 714–721.

[WE07] T. Wang and R. C. Elston, *Improved power by use of a weighted score test for linkage disequilibrium mapping*, Am. J. Hum. Genet **80** (2007), 353–360.

[WGRP99] S. A. Wolfe, H. A. Greisman, E. I. Ramm, and C. O. Pabo, *Analysis of zinc fingers optimized via phage display: evaluating the utility of a recognition code*, J. Mol. Biol **285** (1999), no. 5, 1917–1934.

[WJ06] M. J. Wainwright and M. I. Jordan, *Log-determinant relaxation for approximate inference in discrete markov random fields*, IEEE Transactions on Signal Processing **54** (2006), no. 6, 2099–2109.

[WMS+11] T. Wurschum, H. P. Maurer, B. Schulz, J. Mhring, and J. C. Reif, *Bayesian inference of epistatic interactions in case-control studies*, Theor. Appl. Genet **123** (2011), no. 1, 109–118.

[WYL+05] H. Wang, Y.M.Zhang, X. Li, G. L. Masinde, S. Mohan, D. J. Baylink, and S. Xu, *Bayesian shrinkage estimation of quantitative trait loci parameters*, Genetics **170** (2005), 465–480.

[XLF04] M. Xiong, J. Li, and X. Fang, *Identification of genetic networks*, Genetics **166** (2004), 10371052.

[Xu03] S. Xu, *Estimating polygenic effects using markers of the entire genome*, Genetics **163** (2003), no. 2, 789–801.

[Xu07] _____, *An empirical bayes method for estimating epistatic effects of quantitative trait loci*, Biometrics **63** (2007), no. 4, 513–521.

[Xu10] _____, *An expectation maximization algorithm for the Lasso estimation of quantitative trait locus effects*, Heredity (2010), 1–12.

[YB09] N. Yi and S. Banerjee, *Hierarchical generalized linear models for multiple quantitative trait locus mapping*, Genetics **181** (2009), no. 3, 1101–1113.

[YGA03] N. Yi, V. George, and D. B. Allison, *Stochastic search variable selection for identifying multiple quantitative trait loci*, Genetics **164** (2003), no. 3, 1129–1138.

[YMF+14] Z. Yao, K. L. Macquarrie, A. P. Fong, S. J. Tapscott, W. L. Ruzzo, and R. C. Gentleman, *Discriminative motif analysis of high-throughput dataset*, Bioinformatics **30** (2014), no. 6, 775783.

[YX08] N. Yi and S. Xu, *Bayesian LASSO for quantitative trait loci mapping*, Genetics **179** (2008), no. 2, 1045–1055.

[Zen94] Z. B. Zeng, *Precision mapping of quantitative trait loci*, Genetics **136** (1994), 1457–1468.

[ZG03] Z. Zhang and M. Gerstein, *Of mice and men: phylogenetic footprinting aids the discovery of regulatory elements*, Journal of Biology **2** (2003), no. 2, 11–15.

[ZGD11] F. Zhang, X. Guo, and H. Deng, *Multilocus association testing of quantitative traits based on partial least-squares analysis*, PLoS One **6** (2011), no. 2, e16739.

[ZH05] H. Zou and T. Hastie, *Regularization and variable selection via the elastic net*, J. Roy. Stat. Soc. B. Met **67** (2005), no. 2, 301–320.

[ZL07] Y. Zhang and J. S. Liu, *Bayesian inference of epistatic interactions in case-control studies*, Nat. Genet **39** (2007), 1167–1173.

[ZLL+04] J. Zhu, P. Y. Lum, J. Lamb, D. GuhaThakurta, S. Edwardsa, R. Thieringer, J. P. Berger, M. S. Wu, J. Thompson, A. B. Sachs, and E. E. Schadt, *An integrative genomics approach to the reconstruction of gene networks in segregating populations*, Cytogenet Genome **105** (2004), no. 2-4, 363–374.

[Zor05] C. Zorn, *A solution to separation in binary response models*, Polit. Anal **13** (2005), 157–170.

[ZWZ+07] J. Zhu, M. C. Wiener, C. Zhang, A. Fridman, E. Minch, P. Y. Lum, J. R. Sachs, and E. E. Schadt, *Increasing the power to detect causal associations by combining genotypic and expression data in segregating populations*, Comput Biol **3** (2007), no. 4, e69.

[ZX05] Y. M. Zhang and S. Xu, *A penalized maximum likelihood method for estimating epistatic effects of QTL*, Heredity **95** (2005), 96–104.

[ZZ99] J. Zhu and M. Q. Zhang, *Scpd: a promoter database of the yeast saccharomyces cerevisiae*, Bioinformatics **15** (1999), 607–611.

[ZZS+08] J. Zhu, B. Zhang, E. N. Smith, B. Drees, R. B. Brem, L. Kruglyak, E. R. E. Bumgarner, and E. E. Schadt, *Integrating large-scale functional genomic data to dissect the complexity of yeast regulatory networks*, Nat Genet **40** (2008), no. 7, 854–861.